TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

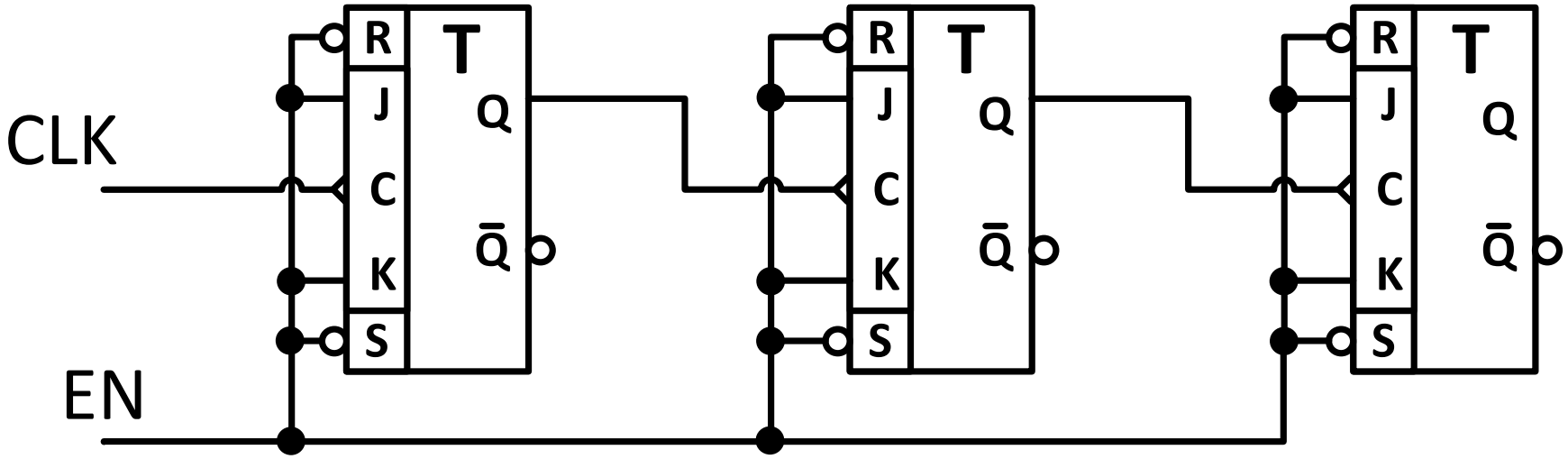
Binární čítače a paměti

Prof. Ing. Ondřej Novák, CSc.
ITE

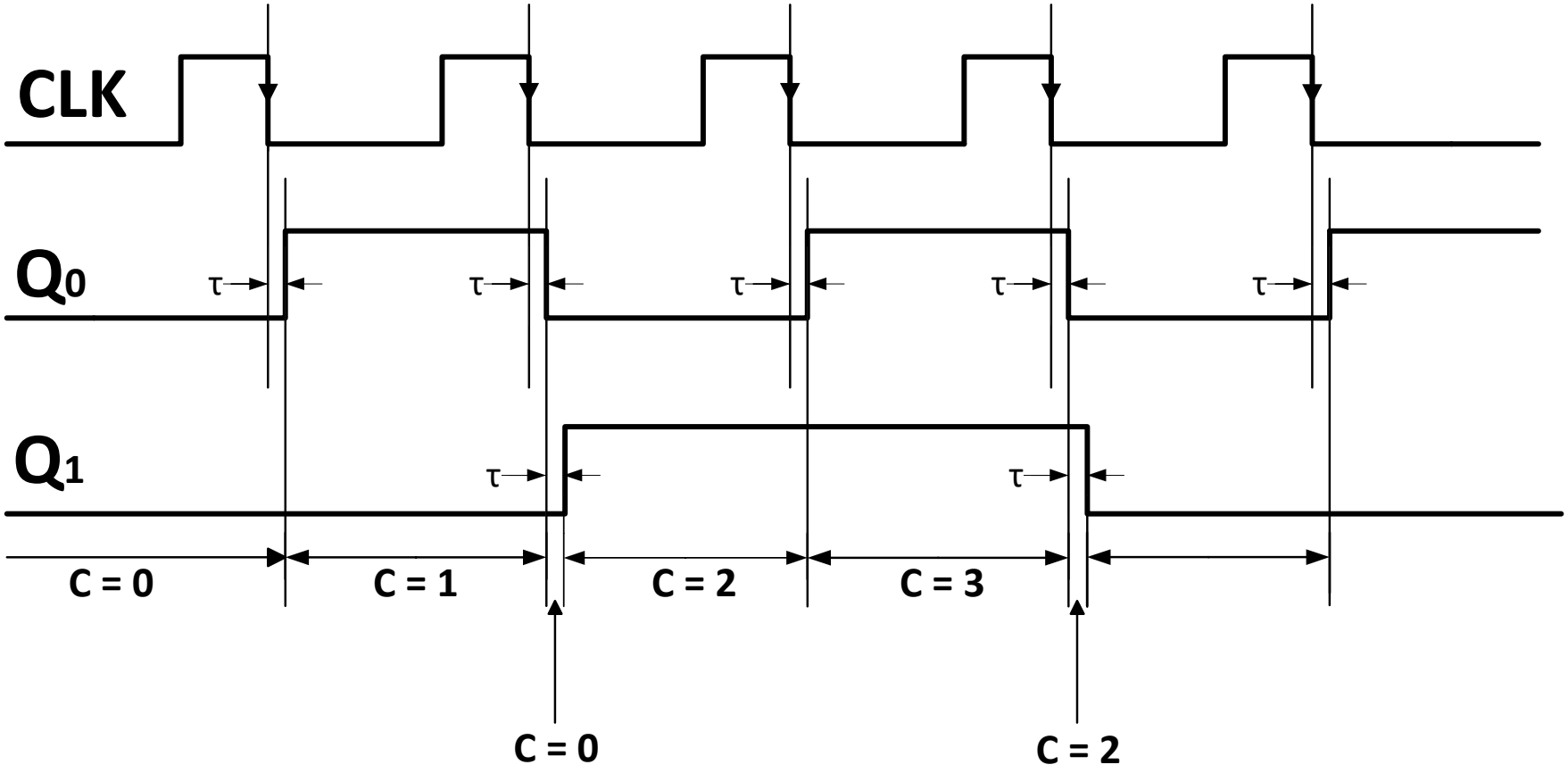
evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Projekt ESF CZ.1.07/2.2.00/28.0050
Modernizace didaktických metod
a inovace výuky technických předmětů.

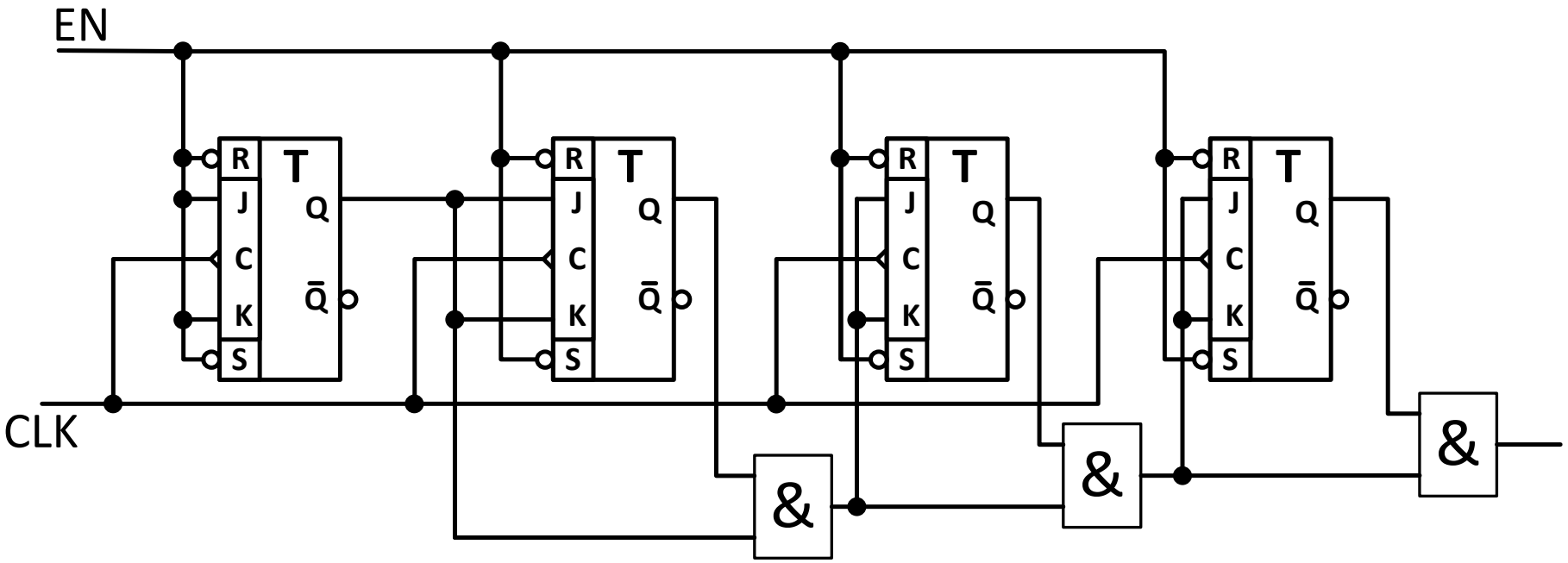
Asynchronní čítač (ripple-carry counter)



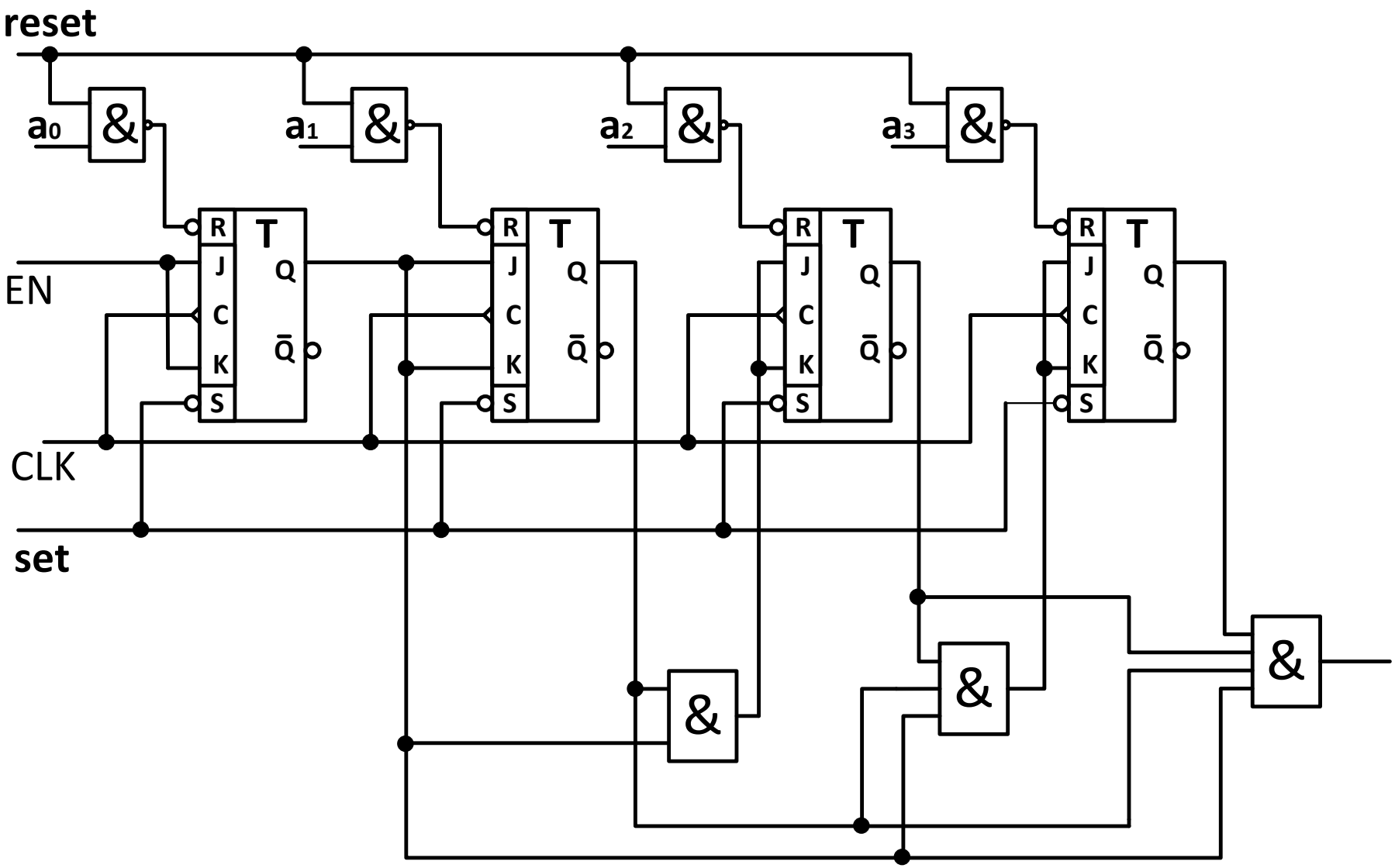
Časový diagram asynchronního čítače



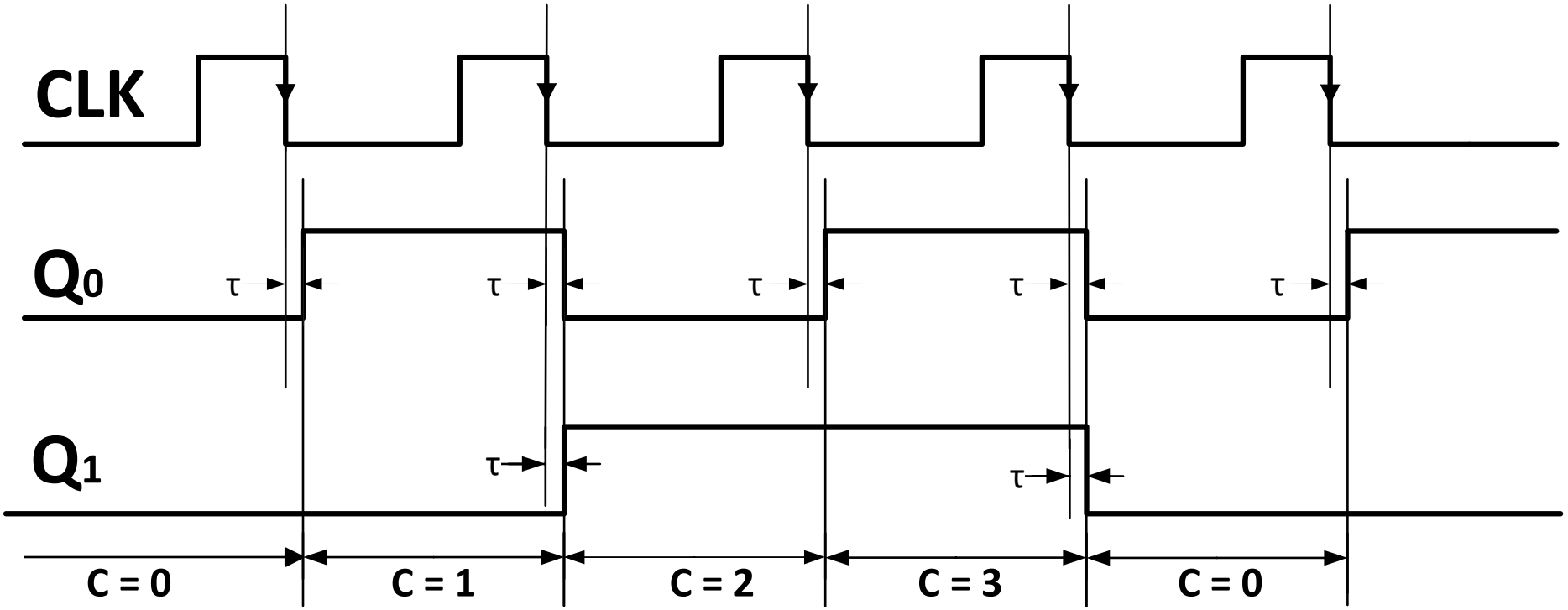
Synchronní čítač se sériovým přenosem



Synchronní čítač s paralelním přenosem



Časový diagram synchronního čítače



Datové typy pro syntézu

- Implicitní celočíselný typ **Integer**
  - Nepoužíváme pro porty nebo signály – pouze vnitřní proměnné (for, generate…)
- ieee.std_logic_1164

  Používáme v **entity**:
  - **std_logic** – ‘U’, ‘X’, ‘0’, ‘1’, ’Z’, ‘W’, ‘L’, ‘H’, ‘-’
  - **std_logic_vector(X downto Y)** – array of std_logic.
- ieee.numeric_std
  - Užíváme v **architecture**:
  - **unsigned(X downto Y)** – array of std_logic;
  - **signed(X downto Y)** – array of std_logic;

‘0’…strong 0
‘1’…strong 1
‘L’…weak 0
‘H’…weak 1
‘Z’…high impedance‘
U’…unknown unasigned)‘
X’…strong undefined (‘0’a ‘1’na jednésítí)
‘W’…weak undefined (‘L’a ‘H’na jedné síti)
‘-’…don’t care

Signed & Unsigned

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity testcase is
end entity;

architecture showcase of testcase is

signal a_slv      : std_logic_vector(3
downto 0);
signal a_unsigned : unsigned(3 downto 0);
signal a_signed   : signed(3 downto 0);

begin

a_slv      <= "1111";
a_unsigned <= "1111";
a_signed   <= "1111";
```

15 – binární váhový kód

-1 – dvojkový doplněk

Silné datové typy

- VHDL má silné datové typy – neexistují implicitní konverze

C

```c
void test() {

char a;
char b;
int c;

// toto přiřazení je v
pořádku
c = a + b;
}
```

VHDL

```vhdl
signal t_a : signed(7 downto 0);
signal t_b : signed(7 downto 0);
signal t_c : signed(31 downto
0);

begin

-- nelze provést kvůli rozdílným
délkám
t_c <= t_a + t_b;
```

Silné datové typy

- VHDL má silné datové typy – neexistují implicitní konverze

C

```c
void test() {

char a;
unsigned char b;
unsigned char c;

// toto přiřazení je v pořádku
c = a + b;
}
```

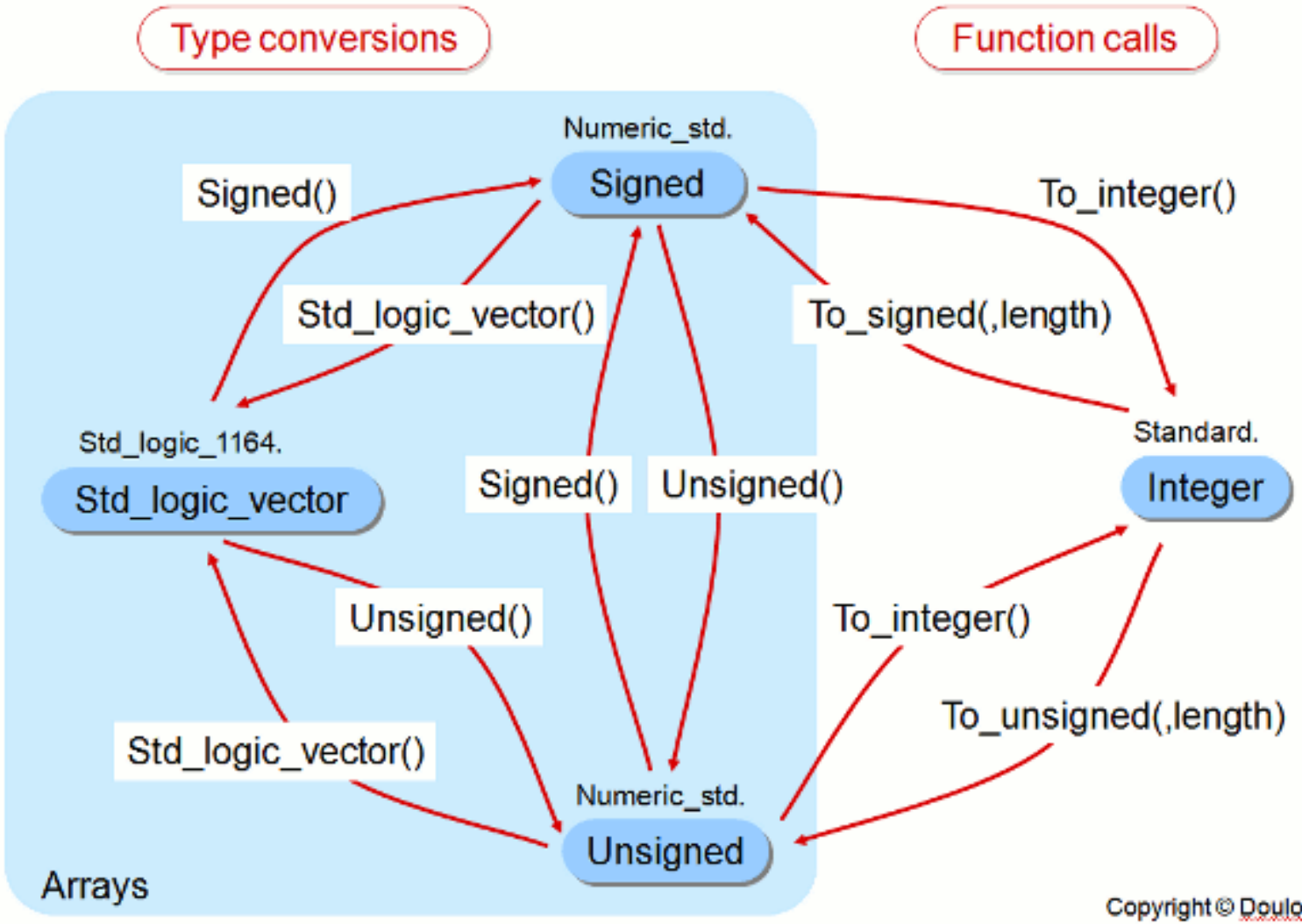
VHDL

```vhdl
signal t_a : signed(7 downto 0);
signal t_b : unsigned(7 downto 0);
signal t_c : signed(7 downto 0);

begin

-- nelze provést kvůli datovým typům
t_c <= t_a + t_b;
```

Povolená přiřazení

- Nelze míchat různé typy polí:
  - **Unsigned = Unsigned** +|-|*|/ **Unsigned**, nebo **Unsigned** +|-|*|/ **Integer**
  - **Signed = Signed** +|-|*|/ **Signed**, nebo **Signed** +|-|*|/ **Integer**
- Lze použít konverzní funkce:
  - **std_logic_vector(unsigned/signed)** – vrací **std_logic_vector** stejné délky
  - **unsigned(std_logic_vector)** – vrací **unsigned** stejné délky
  - **signed(std_logic_vector)** – vrací **signed** stejné délky

Numeric Std Conversions



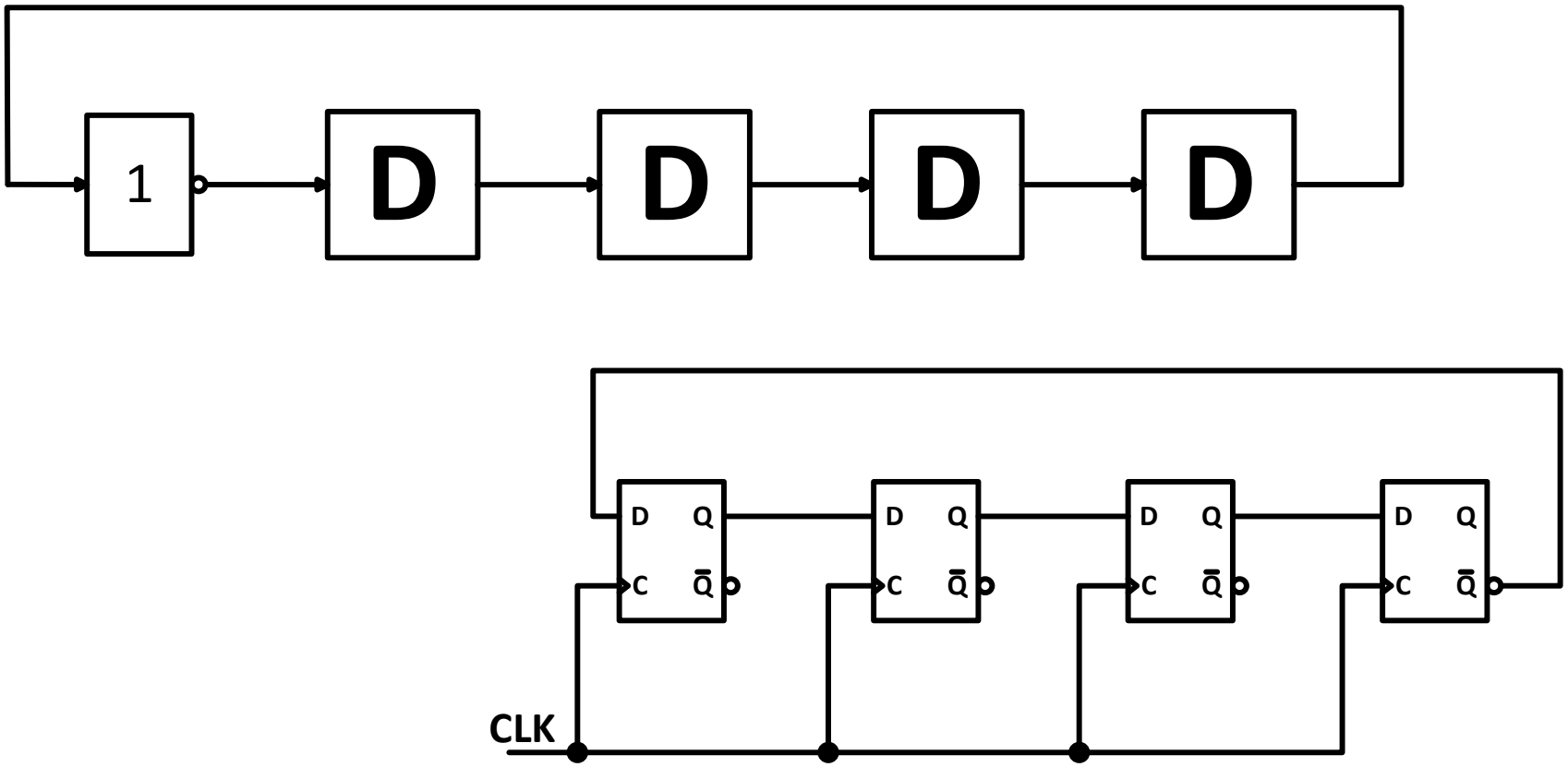
N-bitový generický čítač

```
entity counter is
    generic (
        C_WIDTH : integer := 8
    );
    port (
        clk : in  std_logic;
        rst : in  std_logic;
        ce  : in  std_logic;
        max : in  std_logic_vector(C_WIDTH - 1 downto
        0);
        q   : out std_logic_vector(C_WIDTH - 1 downto
        0)
    );
end entity;
```

Architektura čítače

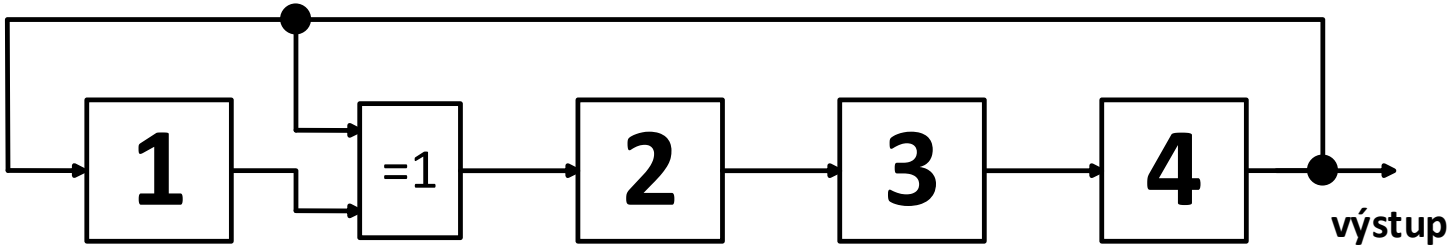
```
architecture RTL of counter is
        signal cnt_reg : unsigned(C_WIDTH - 1 downto 0);
begin
    q <= std_logic_vector(cnt_reg);
    process(clk)
    begin
        if rising_edge(clk) then
            if rst = '1' or cnt_reg >= unsigned(max) then
                cnt_reg <= (others => '0');
            elsif ce = '1' then
                cnt_reg <= cnt_reg + 1;
            end if;
        end if;
    end process;
end architecture;
```

Johnsonův čítač

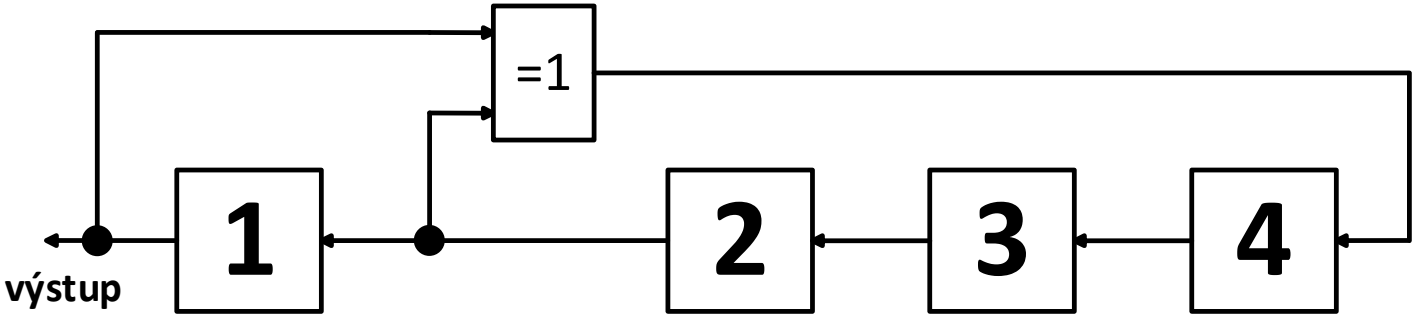


Výhoda: extrémně rychlý
Nevýhoda: nevyužívá všechny možné stavy (využívá 2N stavů)

Lineární zpětnovazební posuvné registry (LFSR)

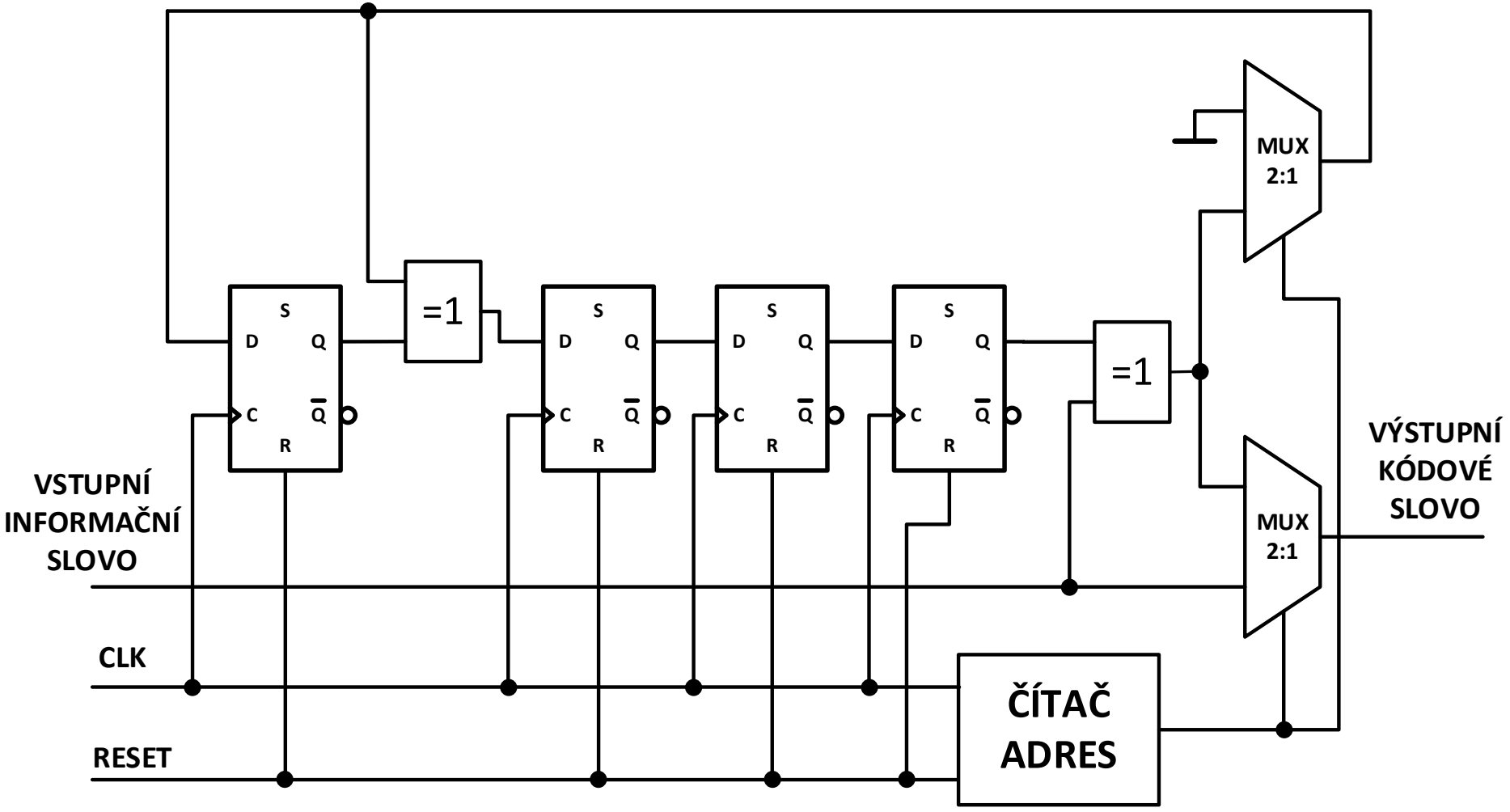


LSFR typ 1

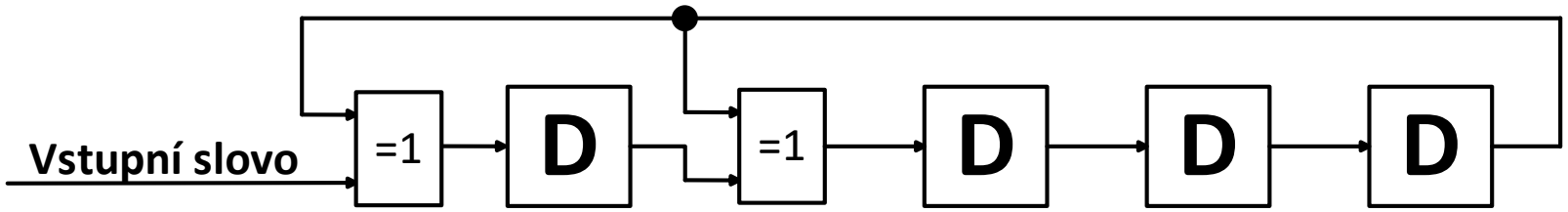


LSFR typ 2

Kodér Hammingova kódu (15,11)



Dekodér (15,11)-kódu



V dekodéru se po přijetí 11 datových bitů a 4 kontrolních bitů vytvoří syndrom. Je-li syndrom roven 0000, nedošlo k chybě přenosu. Při jednochybě je možné na základě analýzy syndromu vadný bit opravit. Analýza trvá 15 taktů. Minimální kódová vzdálenost kódu je 3. Z toho Vyplývá možnost detekce dvojchyby a opravy jednochyby. Pro vyšší kódové vzdálenosti větší možnosti oprav.

Celulární automaty

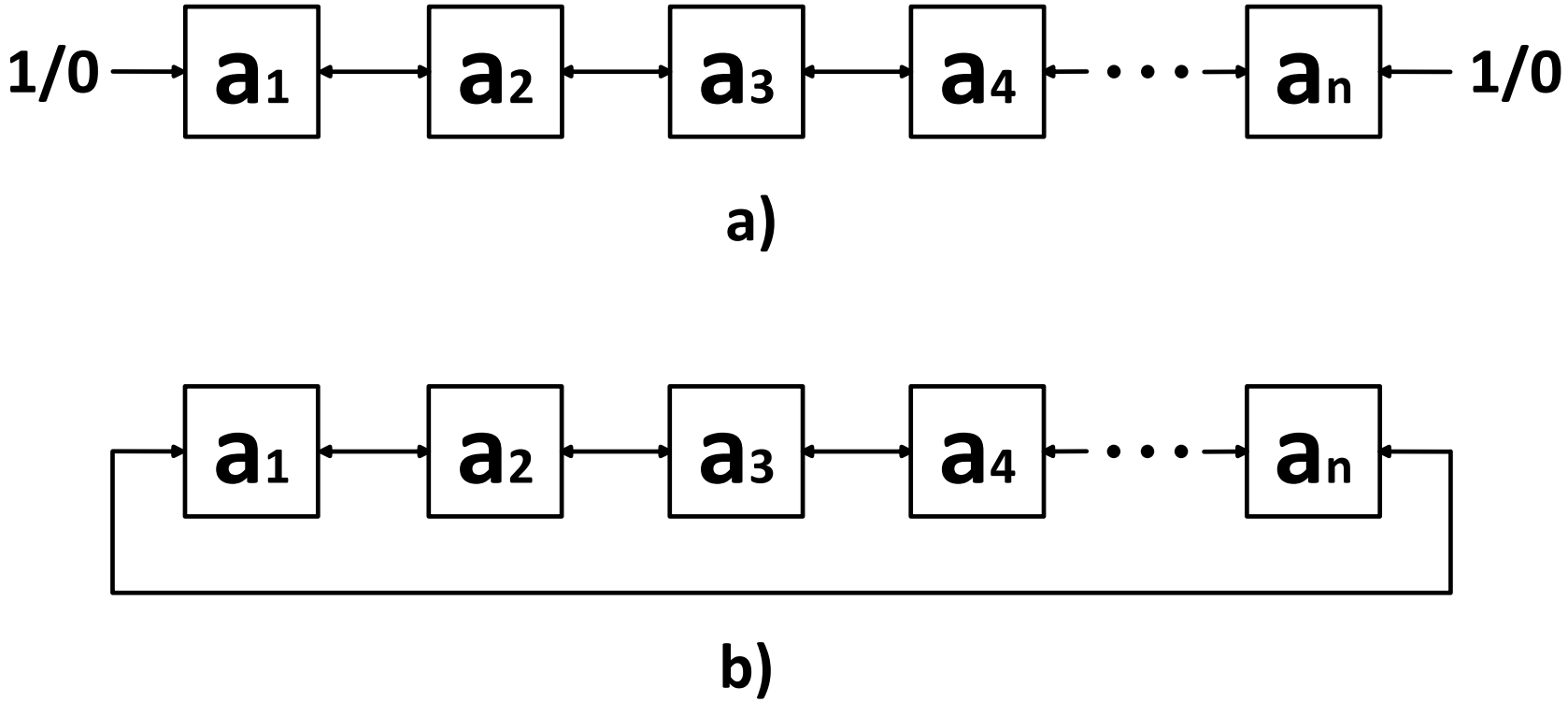
- podmnožina konečných automatů
- jedno a vícerozměrné
- stav buňky vyjádřen jedním nebo více bity
- lineární nebo nelineární

Lokální pravidla tvoření následujícího stavu

Příklad: pravidlo 90:

stav 3 buněk	111	110	101	100	011	010	001	000
následující stav	0	1	0	1	1	0	1	0

Celulární automaty

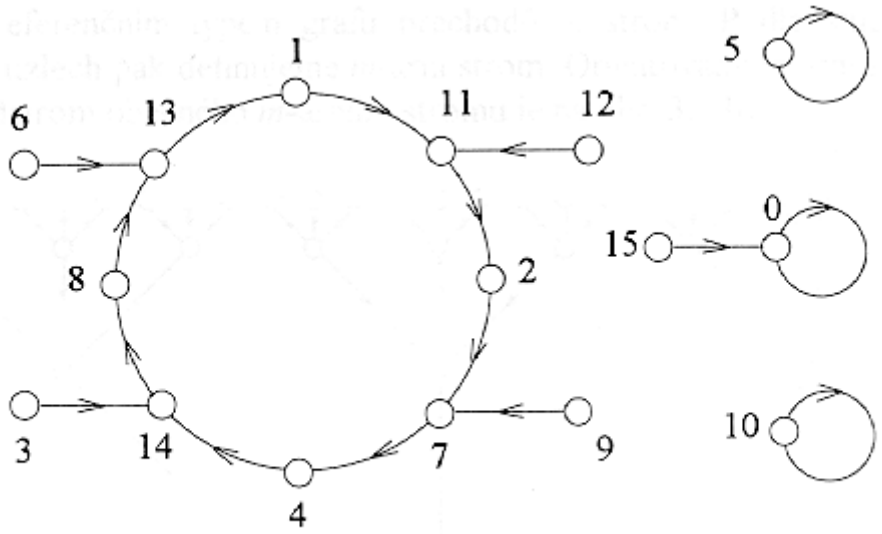


Jednorozměrný CA a) s konstantními okrajovými podmínkami
b) s cyklickými okrajovými podmínkami

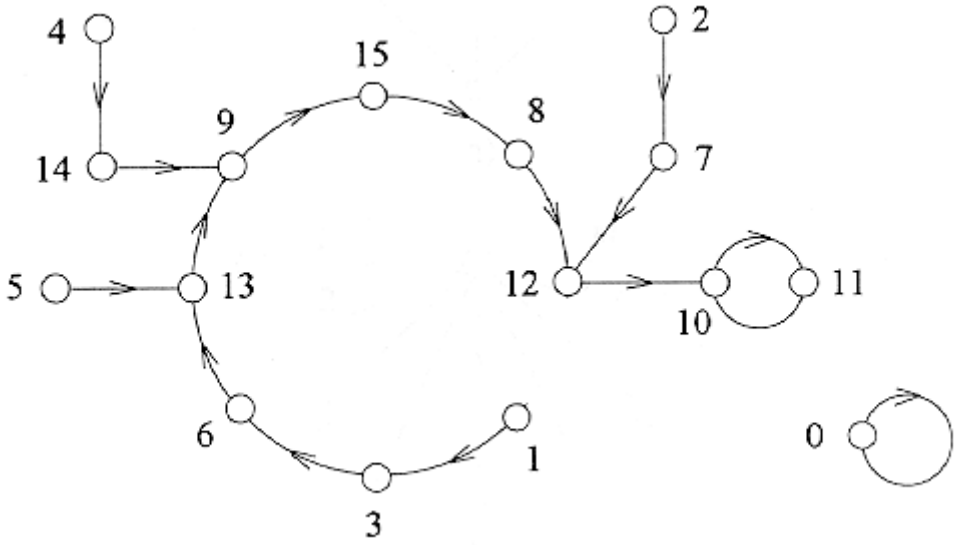
Aditivní pravidla:

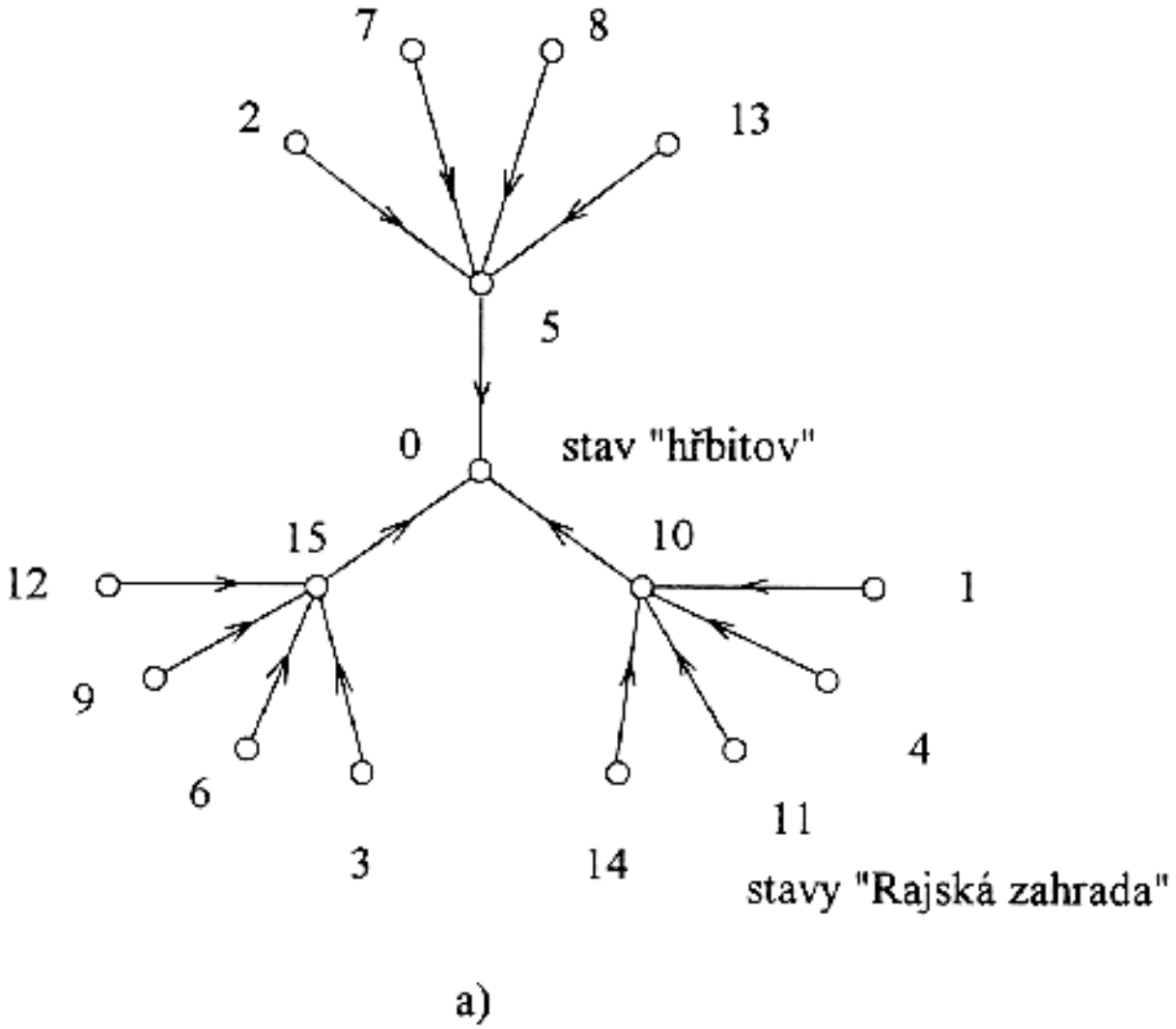
a – aktuální buňka
a_L – levý soused
a_P – pravý soused

Číslo dekadicky	Číslo binárně	Boolovský výraz
0	**00000000**	0
15	**00001111**	$\overline{a_L}$
51	**00110011**	$\overline{a}$
60	**00111100**	$a_L \oplus a$
85	**01010101**	$\overline{a_P}$
90	**01011010**	$a_L \oplus a_P$
102	**01100110**	$a \oplus a_P$
105	**01101001**	$\overline{a_L} \oplus \overline{a} \oplus \overline{a_P}$
150	**10010110**	$a_L \oplus a \oplus a_P$
153	**10011001**	$\overline{a \oplus a_P}$
165	**10100101**	$\overline{a_L \oplus a_P}$
170	**10101010**	a_P
195	**11000011**	$\overline{a_L \oplus a}$
204	**11001100**	a
240	**11110000**	a_L
255	**11111111**	1

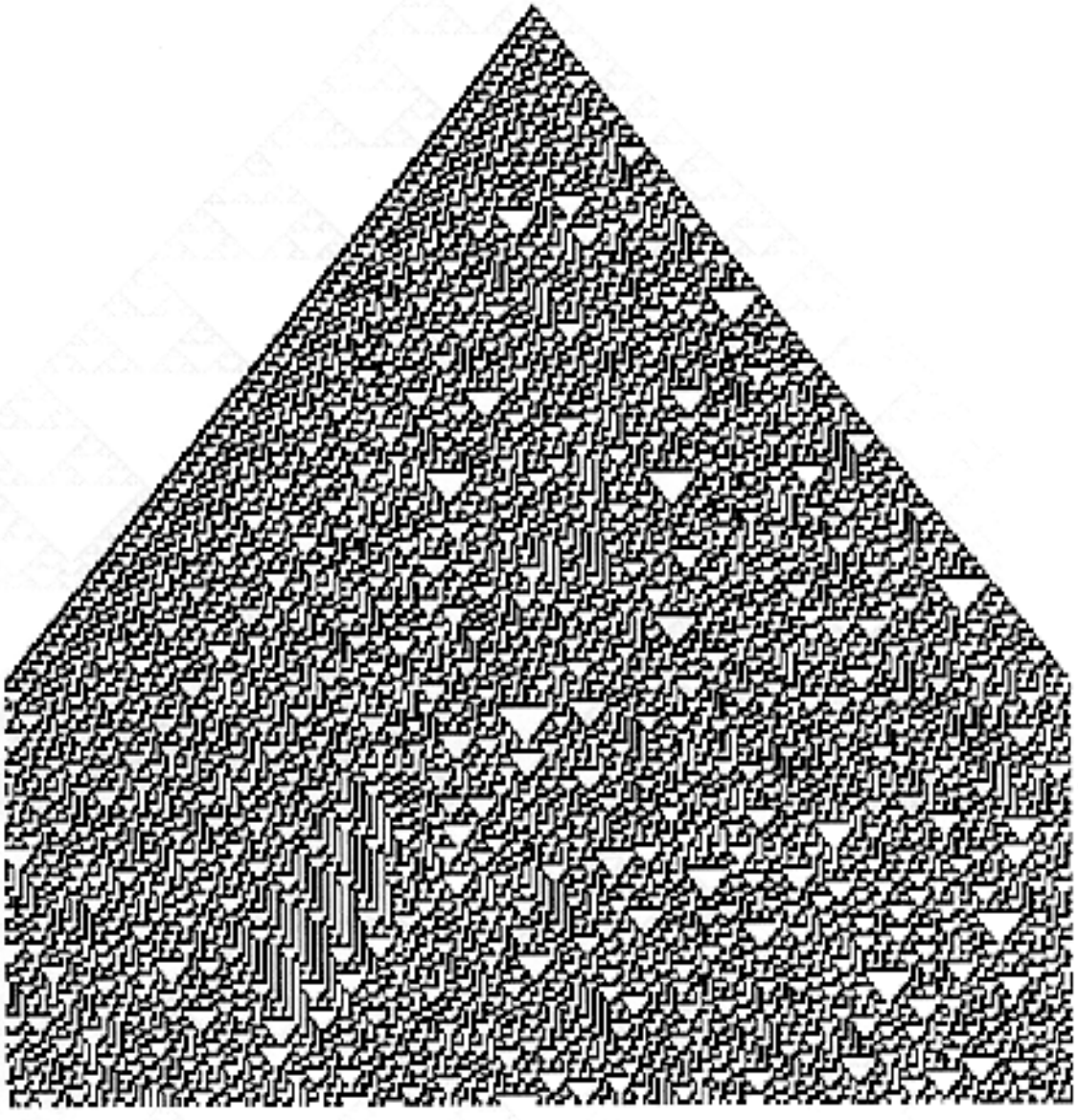


a)




7
8
2
13
5
0
stav "hřbitov"
15
10
12
1
9
4
6
11
3
14
stavy "Rajská zahrada"

a)



Nelineární CA, pravidlo CA 30, n=300, cyklické OP

Dvourozměrné celulární automaty

- Conwayova hra na život (má-li prázdná pozice v sousedství tři buňky, vznikne zde nová buňka, dotýká-li se buňka méně než dvou buněk, buňka zaniká, dotýká-li se více než jedné buňky, buňka se udrží.
- Von Neumannovo 5 bodové sousedství (32 různých možných utvářecích pravidel – vybírají se pouze lineární pravidla
- Moorovo 9 bodové sousedství zahrnuje i sousedy na diagonále
- Výhodné vlastnosti při generování pseudonáhodných čísel.

Paměti

- **rozdělení pamětí**
- **popis pamětí**
- **konstrukční principy a využití v počítačích**

Rozdělení pamětí

podle možností změny dat:

- **obsah lze libovolně přepisovat ..... RAM (Random Acces Memory)**
  - **SRAM ... statická**
  - **DRAM ... dynamická**
- **permanentní**
  - **obsah určen při výrobě .... ROM (Read Only Memory)**
  - **obsah lze jednorázově naprogramovat .... PROM**
- **semipermanentní**
  - **obsah lze naprogramovat a lze vymazat a přeprogramovat .... EPROM, EEPROM, FLASH**
- **volatilní ... energeticky závislé (uložená informace zanikne po vypnutí napájení) .... SRAM, DRAM**
- **nonvolatilní .... ROM, PROM, EPROM, EEPROM, FLASH**

Rozdělení pamětí

podle použití v počítači:

- **hlavní (operační paměť)**
- **vnější paměť**
- **vyrovnávací paměť (cache)**
- **....**

podle fyzikálního principu:

- **polovodičové**
- **magnetické**
- **optické**
- **....**

Moderní paměťové technologie

- **Flash NAND, NOR**
- **MRAM (magnetic RAM), nonvolatilní technologie, omezený počet zápisů**
- **RRAM (Resistive RAM) – změna resistivity vlivem přítomnosti nebo absence vodivých atomů**
- **NRAM (Carbone nanotubes) – změna resistivity po průchodu proudu CN trubičkou**

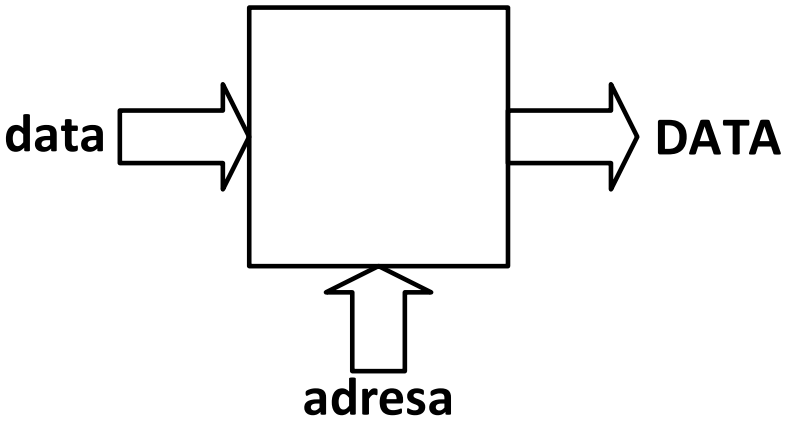
Rozdělení pamětí

podle způsobu výběru položek:

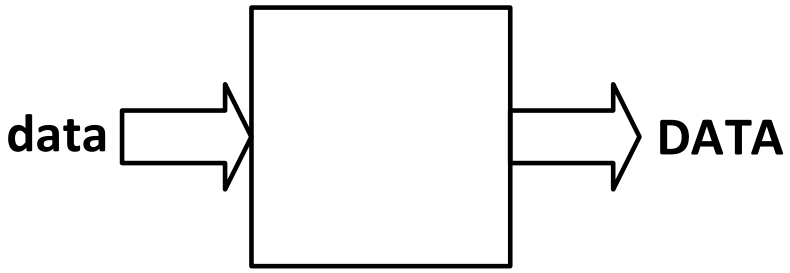
- **s adresovým výběrem (adresové, adresovatelné)**
- **s postupným výběrem (sériové)**
- **asociativní (výběr podle části uložené informace, tzv. klíče ... CAM ... content adressable memory)**
- **zásobník (LIFO ... last in first out)**
- **fronta (FIFO ... first in first out)**

Rozdělení pamětí

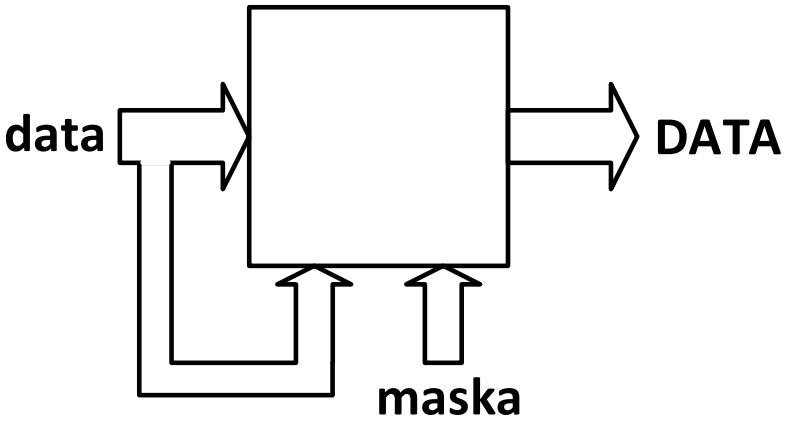
adresovatelná



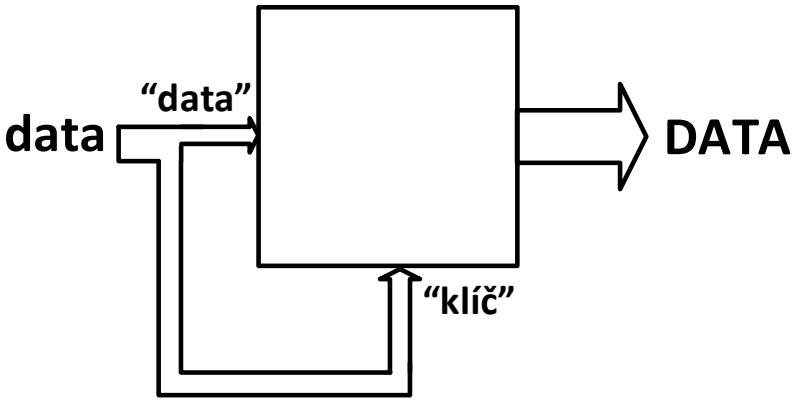
LIFO, FIFO – s postupným výběrem



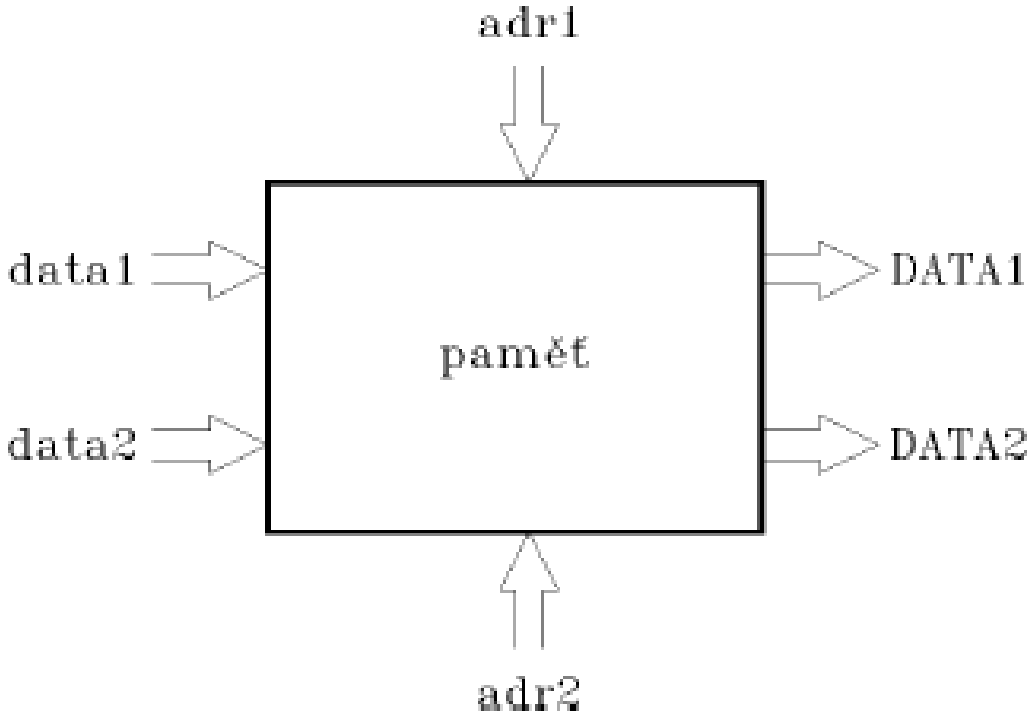
„univerzální“ CAM



„speciální“ CAM



Adresovatelná paměť vícebránová



Základní pojmy

paměťová buňka ... základní stavební blok paměti, slouží k záznamu jednoho bitu

paměťové místo ... skupina paměťových buněk které lze najednou zapisovat nebo číst (šířka slova paměti)

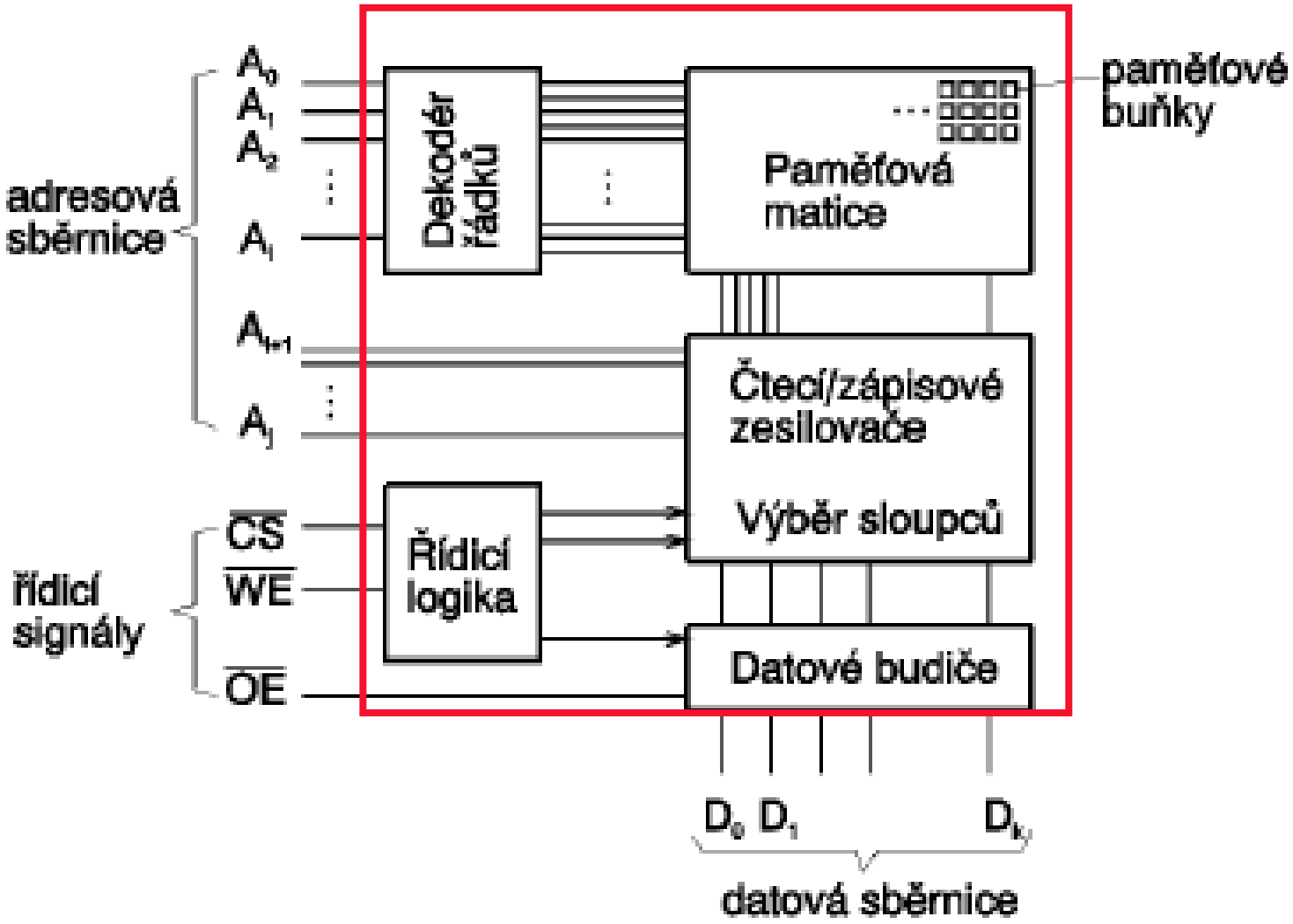
položka ... obsah paměťového místa

adresa ... číselné označení (index) paměťového místa jimž lze vybírat jednotlivé položky

kapacita paměti ... počet položek

paměťová matice ... skupina paměťových míst uspořádaná tak, že je lze vybírat adresou

Blokové schéma typického paměťového obvodu



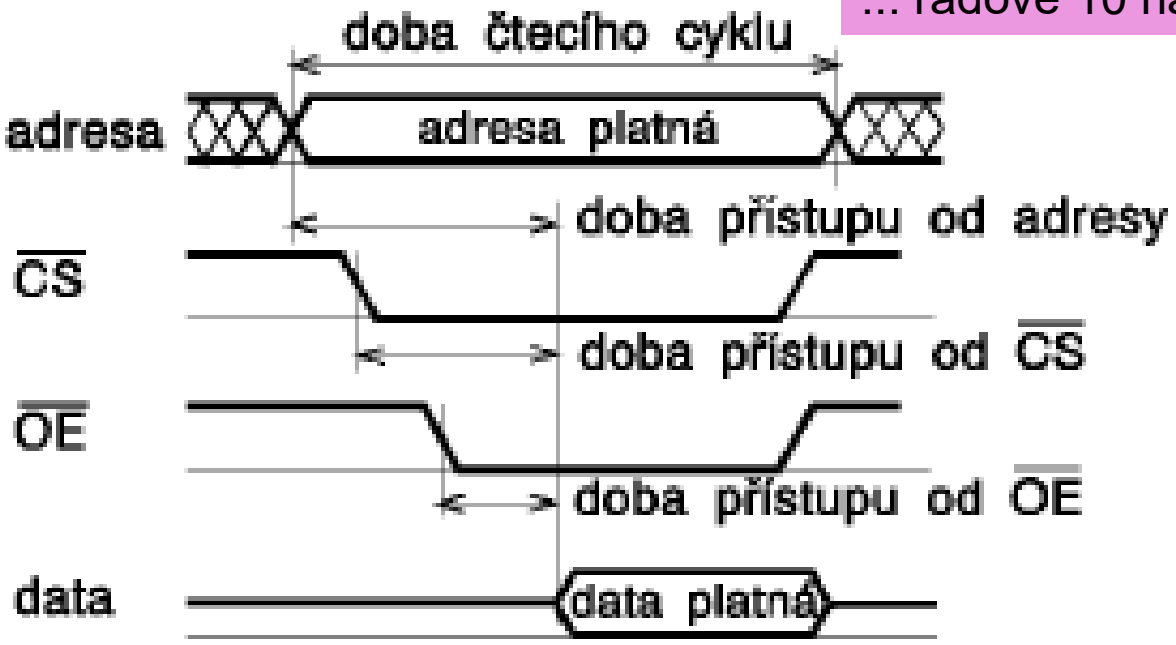
Řídicí signály

- $\overline{\mathbf{OE}}$ - **o**utput **e**nable ... aktivace výstupních třístavových budičů datové sběrnice
- $\overline{\mathbf{WE}}$ - **w**rite **e**nable ... povolení zápisu
- $\overline{\mathbf{CS}}$ - **c**hip **s**elect ... výběr čipu - podmiňuje provedení zápisu nebo čtení

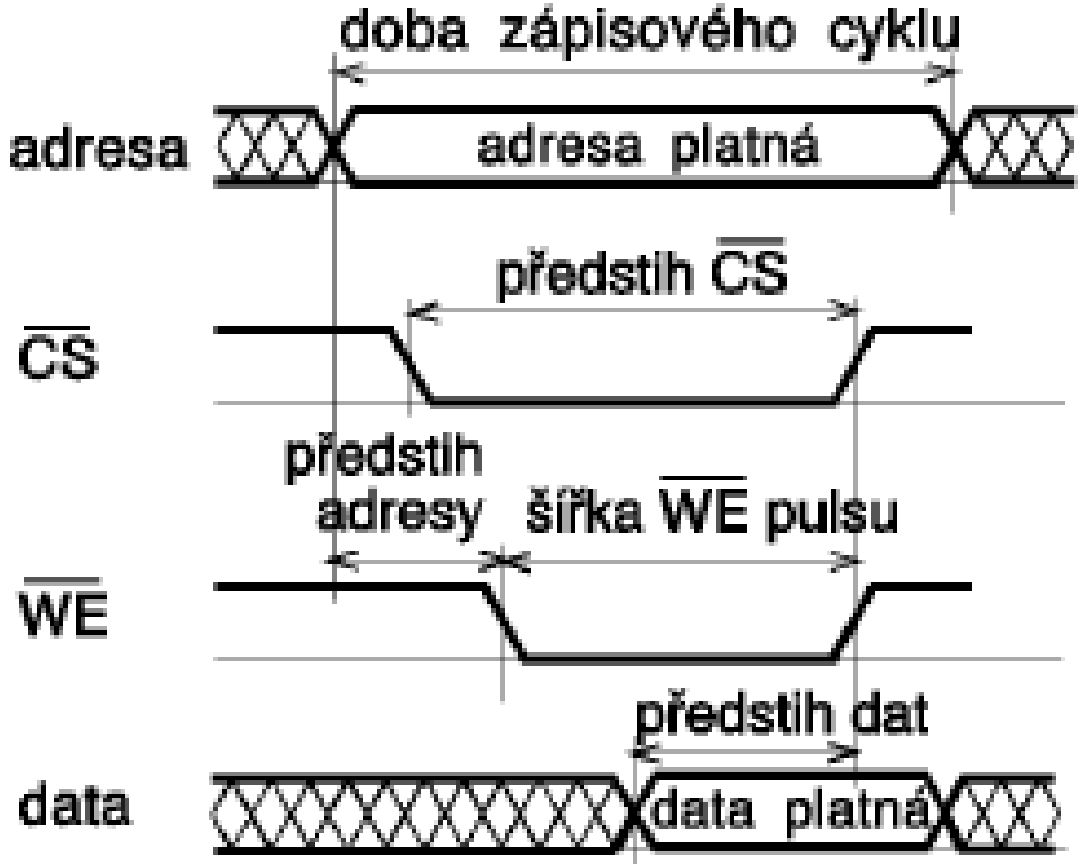
Chování paměťových obvodů, parametry ..... čtení

Typický průběh signálů pro operaci čtení ze statické paměti SRAM (je nutné dodržet minimální nebo i maximální zpoždění mezi aktivací jednotlivých signálů a dalšími událostmi)

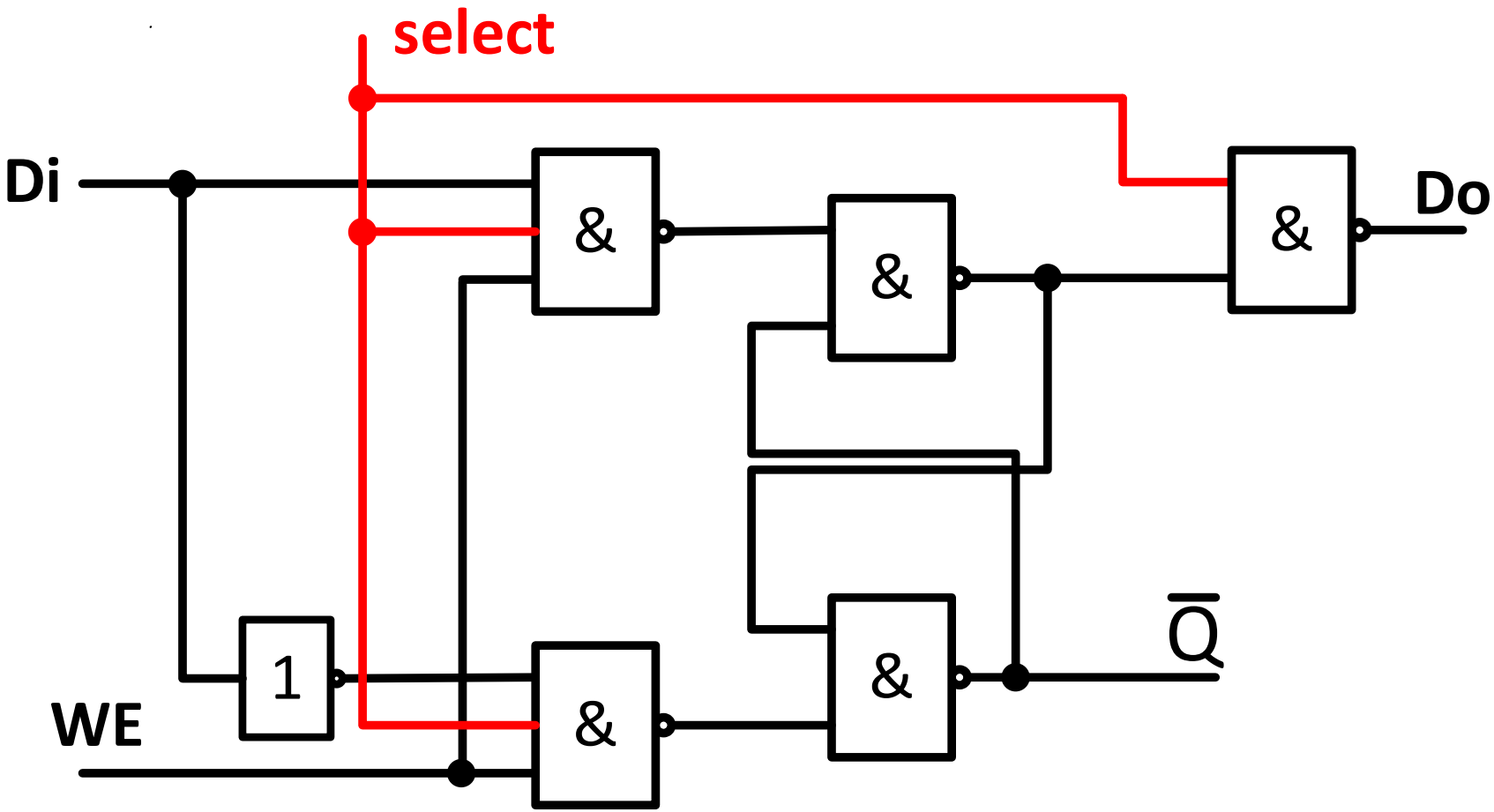
... řádově 10 nanosekund



Chování paměťových obvodů, parametry ..... zápis



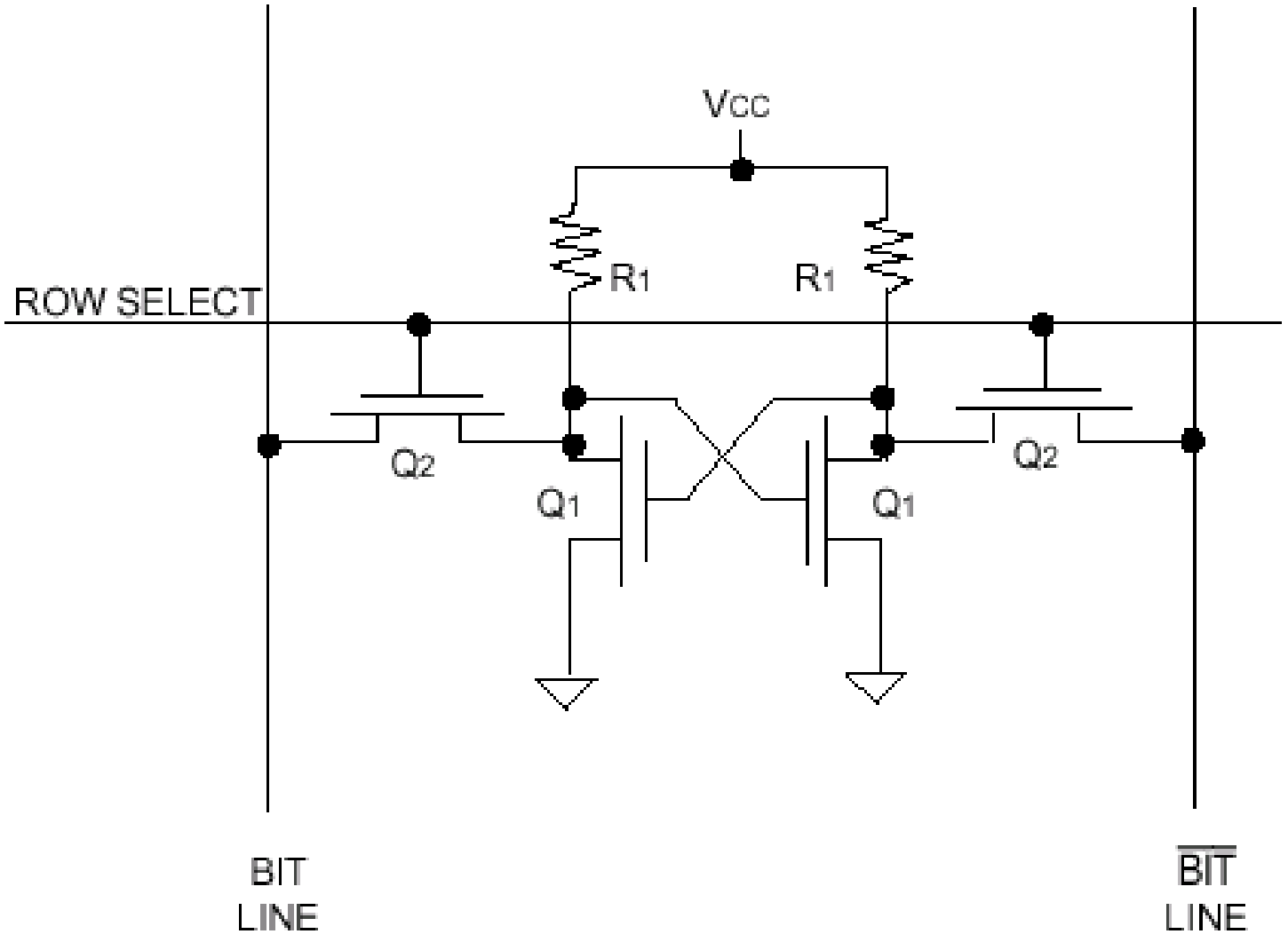
Statická RAM - základní princip



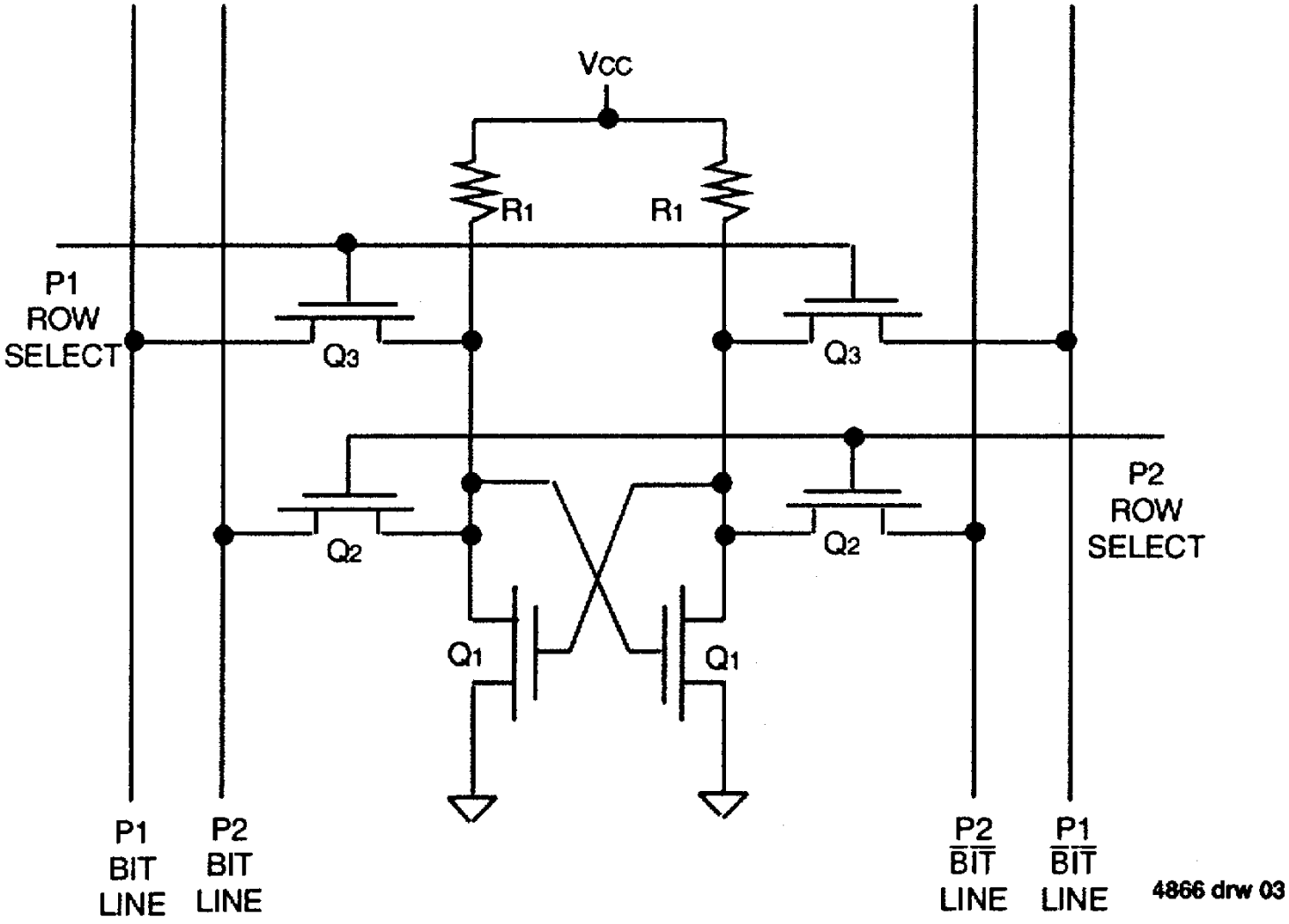
Statické paměti

- Synchronní x asynchronní
- Jednoportové x víceportové

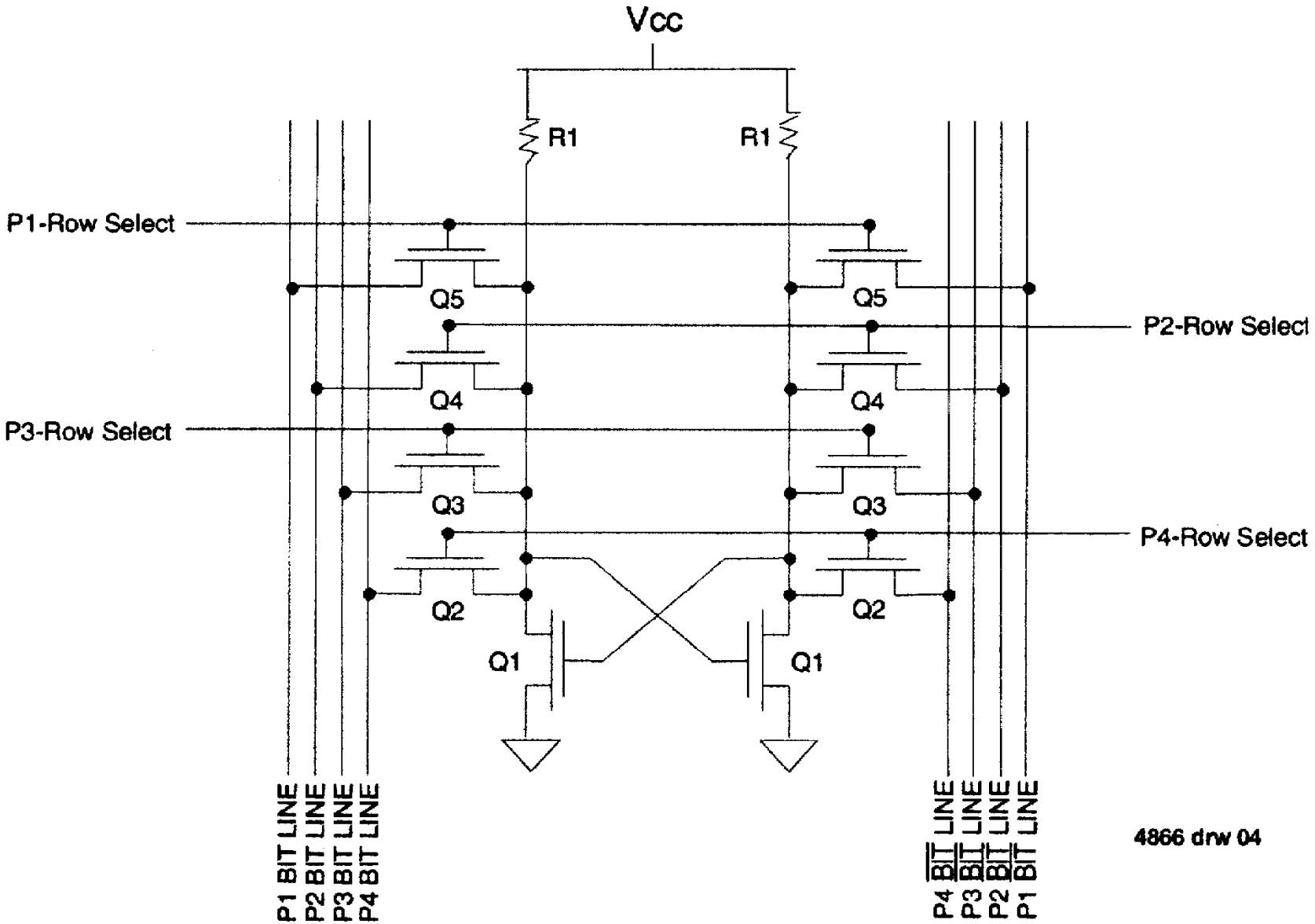
Standardní 2 transistorová statická buňka



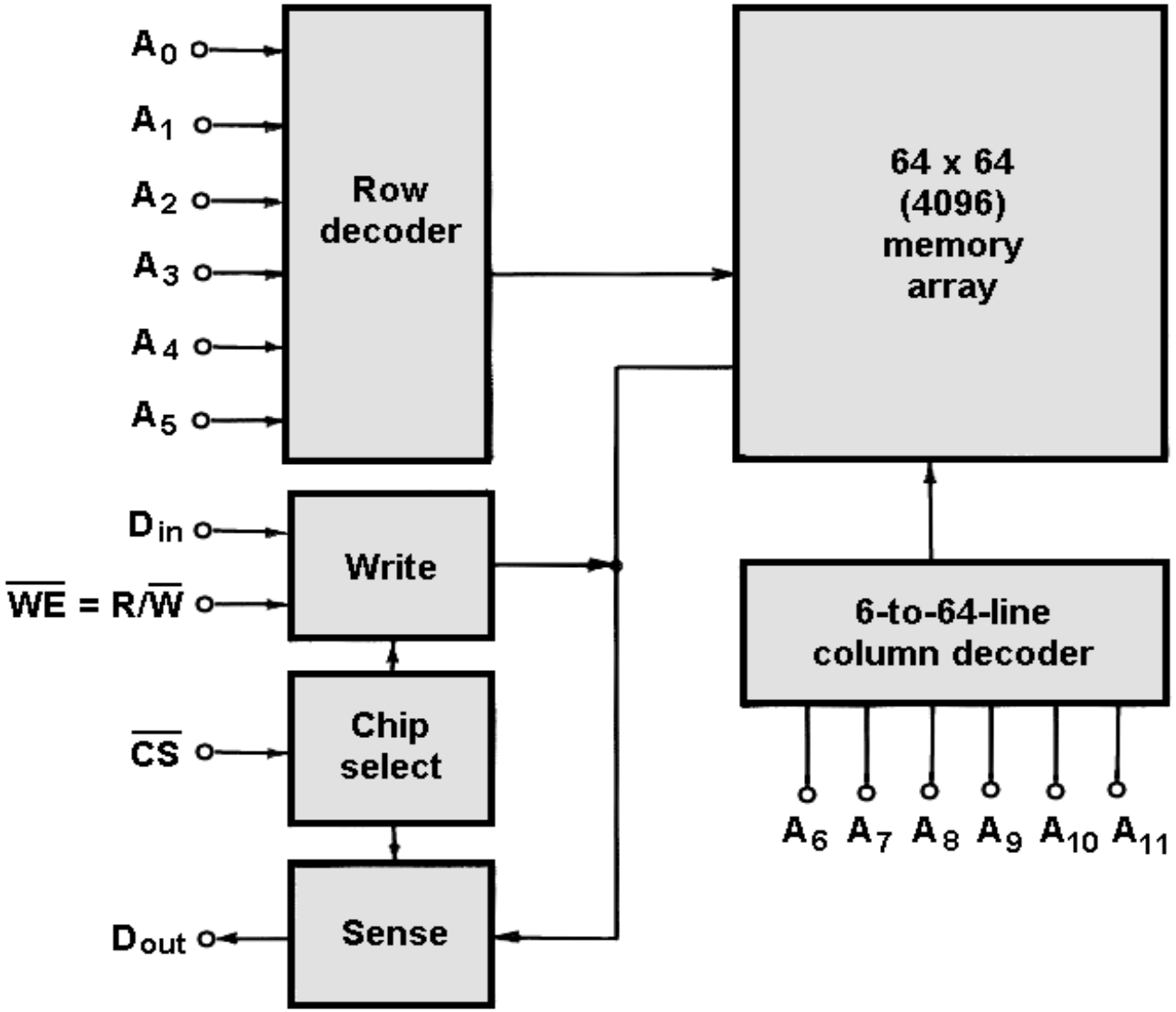
Dvouportová paměťová buňka umožní současný přístup 2 různých zařízení do paměti



Čtyřportová paměťová buňka – čtyřnásobný přístup do paměti

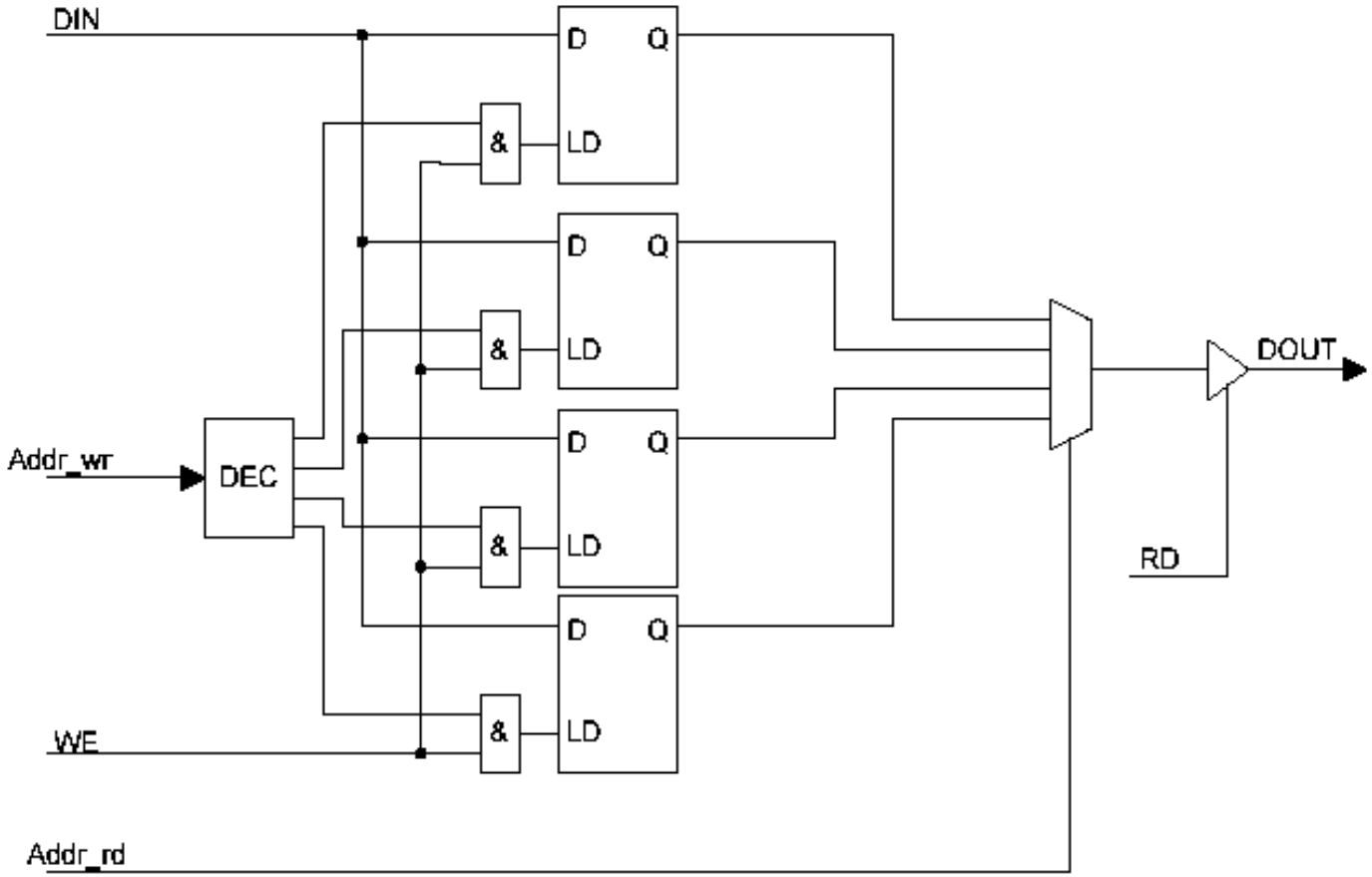


Uspořádání paměti



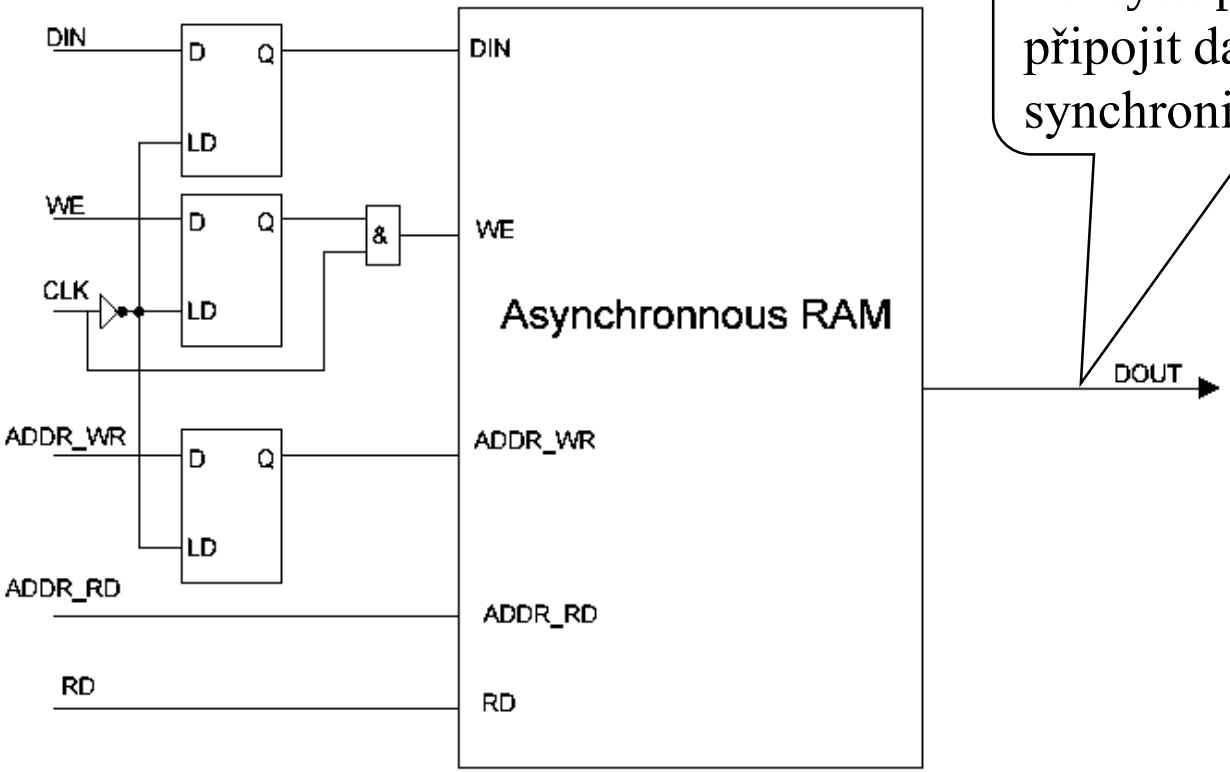
Realizace asynchronní RAM

- pole hladinově řízených klopných obvodů LATCH (malá plocha)
- pro čtení se chová jako kombinační logika
- obtížnější generování řídicích signálů



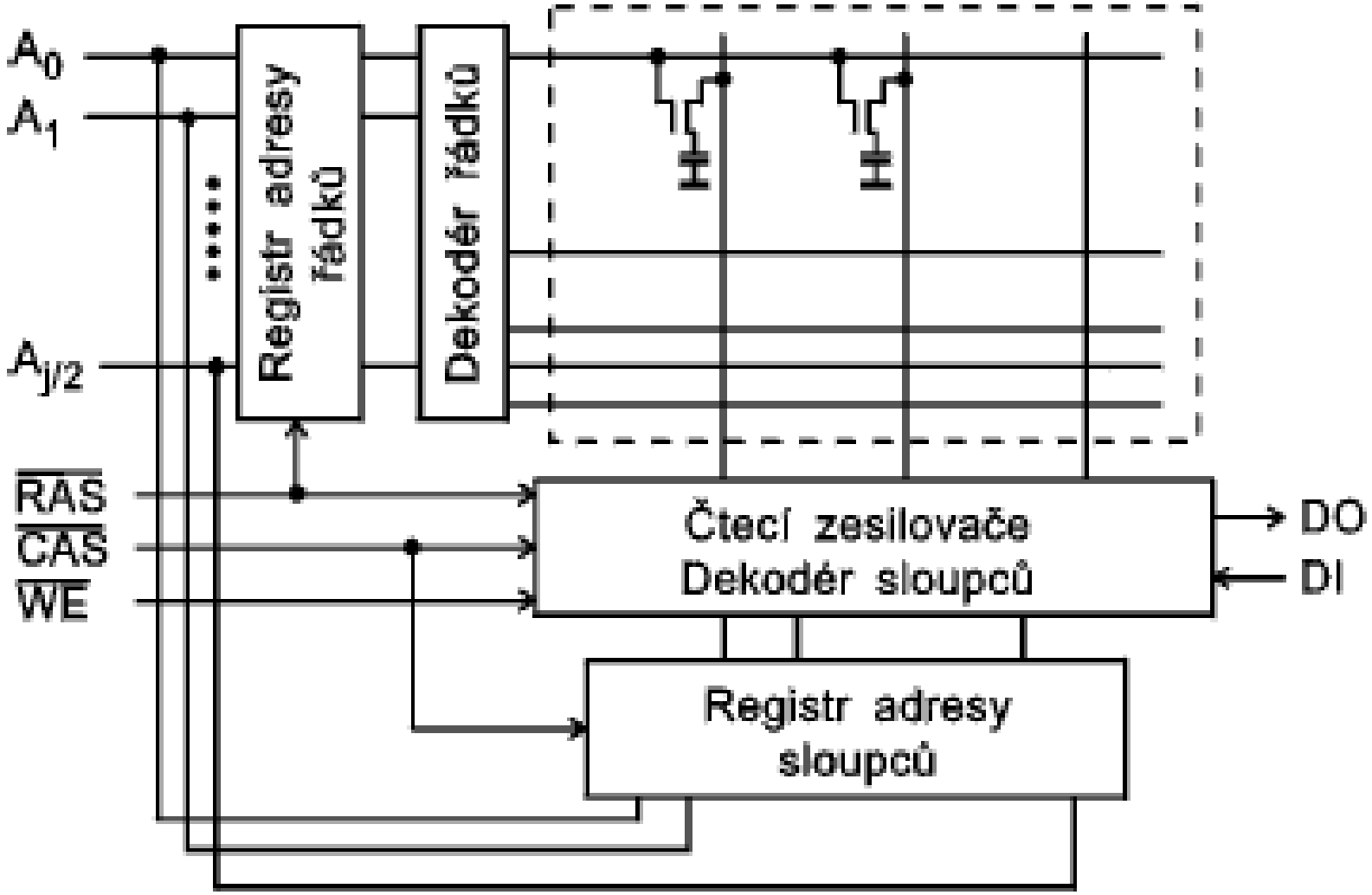
Realizace synchronní RAM pomocí asynchronní RAM

- Pole hladinově řízených klopných obvodů DFF
- DIN, ADDR_WR a WE synchronizovány pomocí CLK
- pro čtení se chová jako kombinační logika



Na výstup možno připojit další synchronizační DFF

Konstrukční princip DRAM



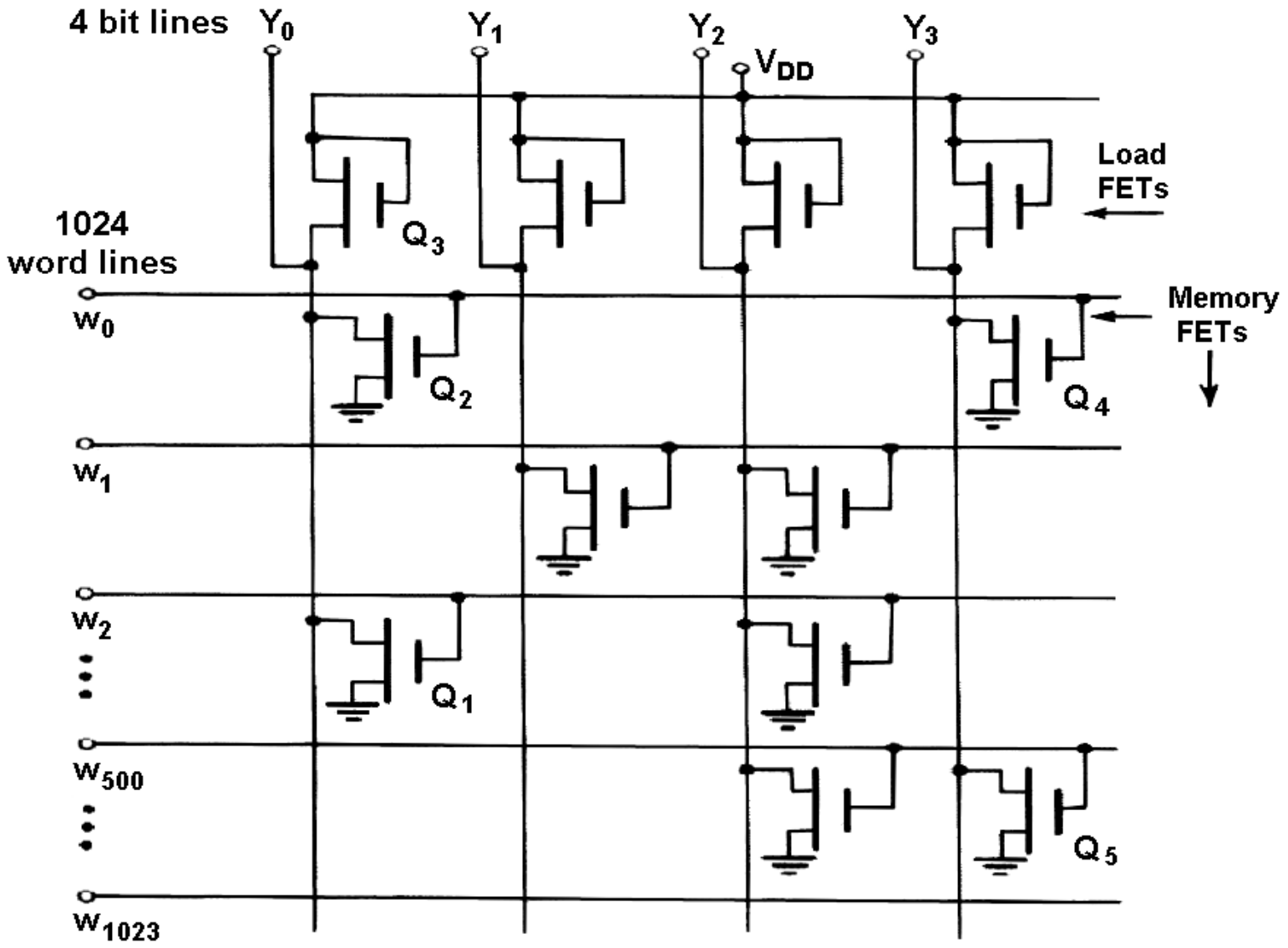
Konstrukční princip DRAM

- Adresa je časově multiplexována, polovina adresy při $\overline{RAS} = 0$ (řádek), druhá polovina adresy při $\overline{CAS} = 0$ (sloupec)
- **Zápis:** Na datový (sloupcový) vodič se přivede zapisovaná úroveň a aktivuje se zvolený řádek. Paměťový kondenzátor se nabije, nebo vybije.
- **Čtení:** Při výběru řádku se kondenzátory vybíjí do vstupů čtecích zesilovačů (čtení je destruktivní, přečtenou informaci je nutno bezprostředně zapsat zpět).
- **Obnovení:** Stejně jako čtení. Všechny čtecí zesilovače jsou umístěny ve všech sloupcích, obnovují se všechny sloupce jednoho řádku najednou.

Porovnání vlastností SRAM a DRAM

- **cena** .... DRAM je při stejné kapacitě levnější než SRAM (1 transistor x 6 transistorů na paměťovou buňku)
- pro refresh (**obnovení**) potřebují DRAM další obvody (periodické generování adres řádků)
- čtení DRAM je **destruktivní**, po čtení musí být informace znovu zapsána ... delší čtecí cyklus
- DRAM mají větší **spotřebu** v klidovém stavu (obnovování)
- (mýtus **rychlosti** ... závisí hlavně na velikosti paměti)

Paměť ROM


4 bit lines
Y_0
Y_1
Y_2
Y_3
V_{DD}
Load FETs
1024 word lines
Q_3
W_0
Q_2
Q_4
Memory FETs
W_1
W_2
Q_1
W_{500}
Q_5
W_{1023}

Synchronní paměť, entita

```
entity sync_mem is
        generic(
                C_WIDTH    : integer := 8;
                C_SIZE     : integer := 1024;
                C_SIZE_LOG : integer := 10
        );
        port(
                clk      : in  std_logic;
                data_in  : in
std_logic_vector(C_WIDTH - 1 downto 0);
                address  : in
std_logic_vector(C_SIZE_LOG - 1 downto 0);
                data_out : out
std_logic_vector(C_WIDTH - 1 downto 0)
        );
end entity;
```

Synchronní paměť, deklarace

```
architecture RTL of sync_mem is

-- definice typu
type ram_t is array (C_SIZE - 1 downto 0) of
std_logic_vector(C_WIDTH - 1 downto 0);
-- deklarace ram
signal ram_inst : ram_t;
-- deklarace registru adresy
signal addr_reg : unsigned(C_SIZE_LOG - 1 downto 0);

begin
```

Synchronní paměť, tělo

```
process(clk)
begin
    if rising_edge(clk) then
        ram_inst(to_integer(unsigned(address))) <= data_in;
        addr_reg <= unsigned(address);
    end if;
end process;

data_out <= ram_inst(to_integer(addr_reg));
```

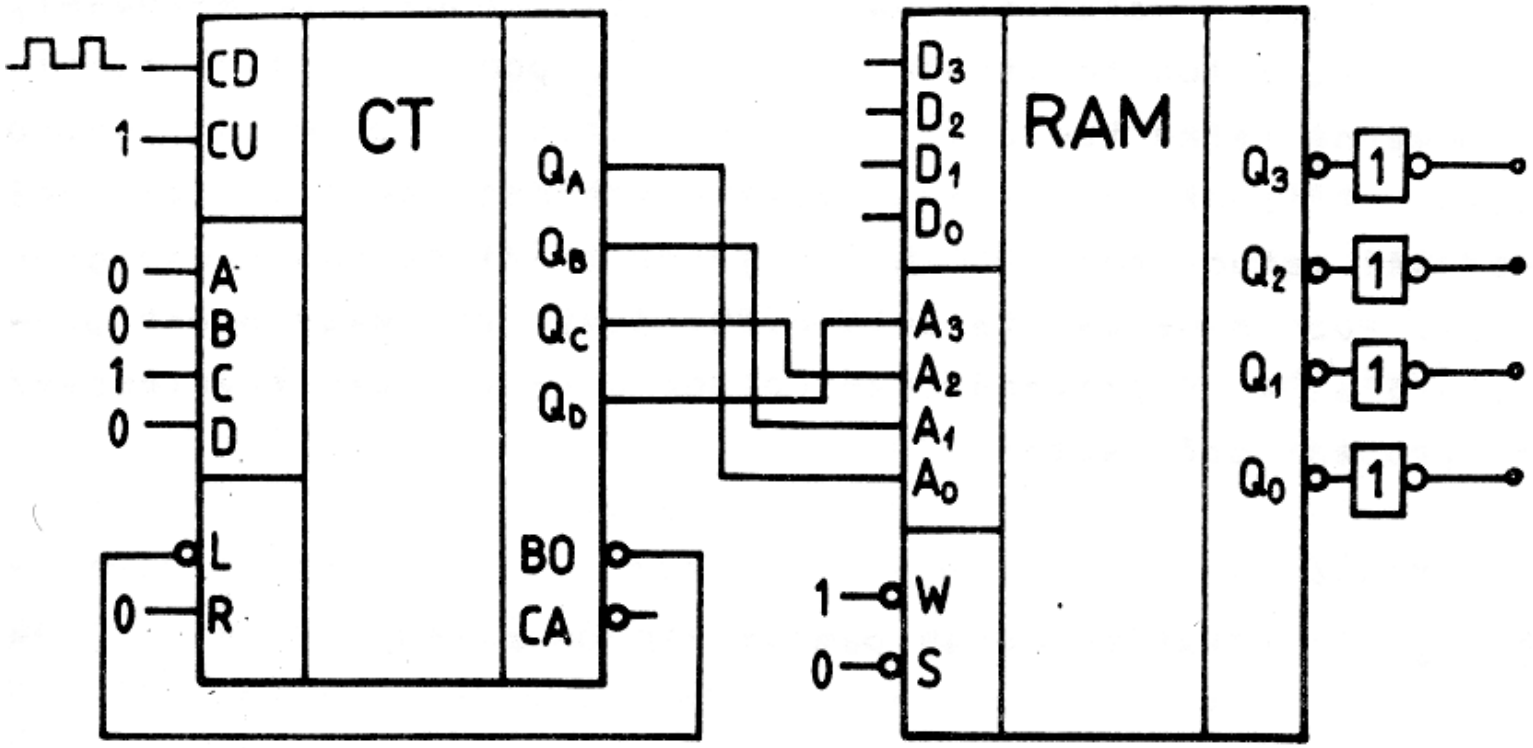
Paměť strukturně

Popis ve VHDL negarantuje technologickou implementaci
V praxi se využívají buď IP makra, nebo IP jádra

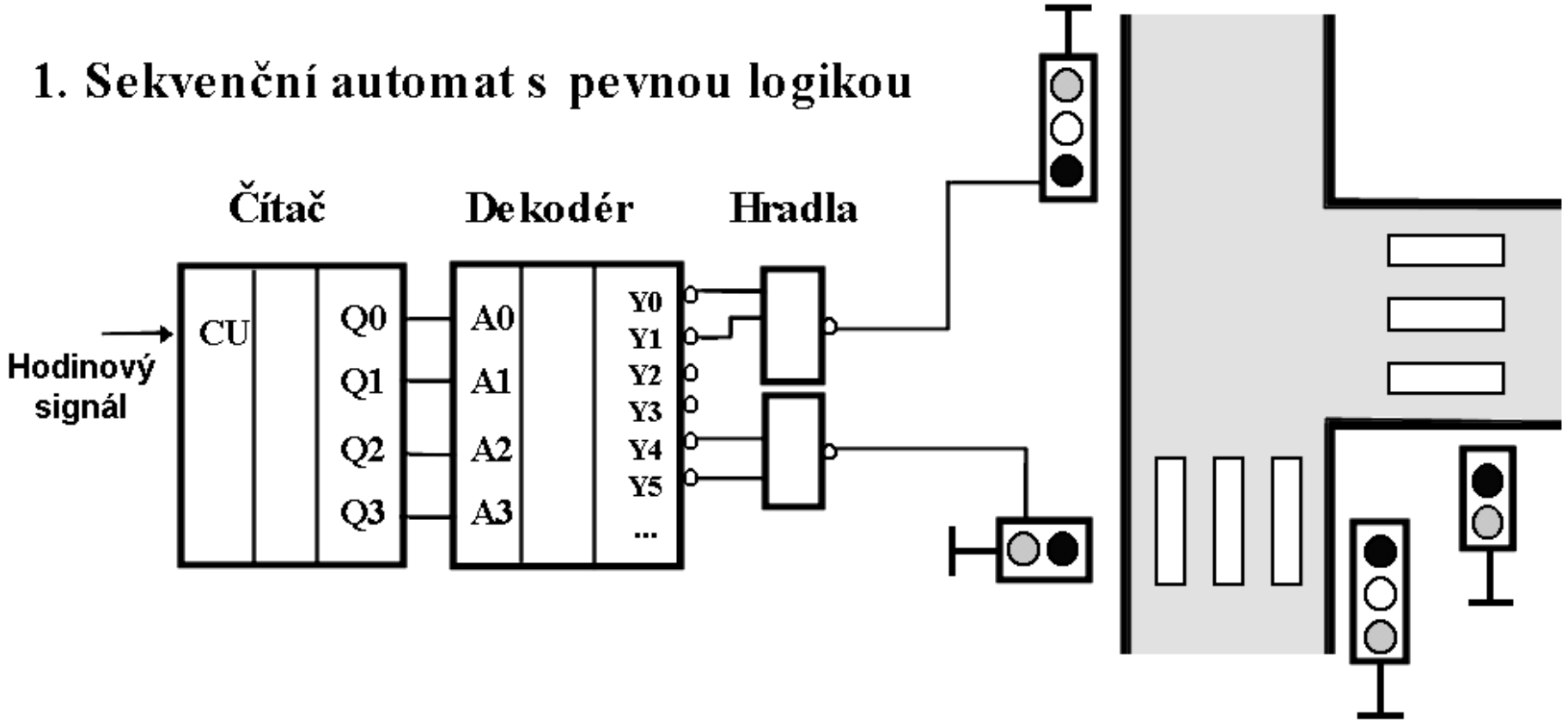
IP makro
- VHDL knihovní prvek
- Dodavatel zaručuje daný typ technologie
- Je do určité míry otevřený – lze modifikovat v instančním kódu

IP jádro
- Dodáváno jako zašifrovaný netlist
- 100% jistota funkce na dané technologii
- Je uzavřený, většinou lze modifikovat pouze pomocí skriptů

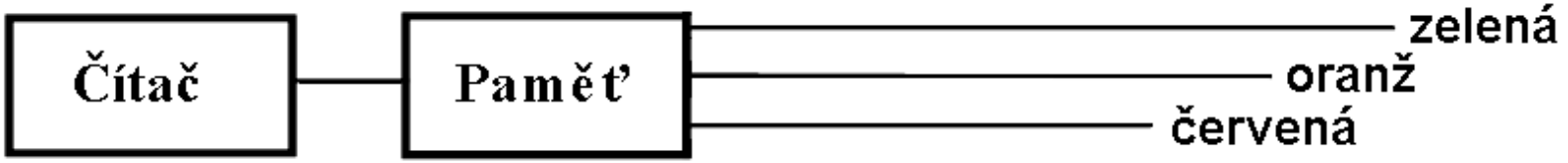
Spojení čítače a paměti



Od sekvenčních automatů k mikroprocesorům



2. Sekvenční automat s pamětí



3. Řídicí systém s mikroprocesorem